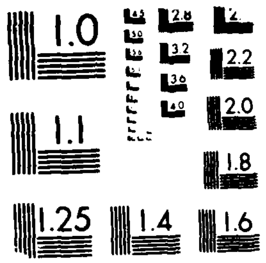
AD-A191 463 KNOWLEDGE BASED CONCEPTS AND ARTIFICIAL INTELLIGENCE: APPLICATIONS TO GUIDANCE AND CONTROL(U) ADVISORY GROUP FOR AEROSPACE RESEARCH AND DEVELOPMENT NEUILLY... 1/2

UNCLASSIFIED AUG 87 AGARD-LS-155 F/G 1/4 NL



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS 1963 A

(1)

AGARD-LS-155

AGARD-LS-155

AGARD LECTURE SERIES No. 155

Knowledge Based Concepts and Artificial Intelligence: Applications to Guidance and Control

DISTRIBUTION AND AVAILABILITY
ON BACK COVER

AGARD-LS-155

NORTH ATLANTIC TREATY ORGANIZATION

ADVISORY GROUP FOR AEROSPACE RESEARCH AND DEVELOPMENT

(ORGANISATION DU TRAITE DE L'ATLANTIQUE NORD)

AGARD Lecture Series No.155

KNOWLEDGE BASED CONCEPTS AND ARTIFICIAL INTELLIGENCE: APPLICATIONS TO GUIDANCE AND CONTROL

This material in this publication was assembled to support a Lecture Series under the sponsorship of the Guidance and Control Panel and the Consultant and Exchange Programme of AGARD presented on 10—11 September 1987 in Ottawa, Canada, 14—15 September in Monterey, USA, 12—13 October 1987 in Delft, The Netherlands and 15—16 October 1987 in Lisbon, Portugal.

Published August 1987

Copyright © AGARD 1987
All Rights Reserved

ISBN 92-835-1588-7

Printed by Specialised Printing Services Limited
40 Chigwell Lane, Loughton, Essex IG10 3TZ

LIST OF SPEAKERS

Lecture Series Director: Mr R.P.Quinlivan
Consultant Engineer Flight Control
General Electric Company
Binghampton NY 13902
USA

SPEAKERS

Mr J.P.Aubert
Laboratoires de Marcoussis
Centre de Recherches de la CGE
Route de Nozay
91460 Marcoussis
France

Dr M.Bird
Lear Seigler Inc.
Instrument and Avionics Systems Div.
M/S 128
4141 Eastern Avenue S.E.
Grand Rapids, MI 49508
USA

Dr A.Bowen
Carlton University
Colonel By Drive
Ottawa
Canada K1S 5B6

Dr B.Ellis
RAE
R177 Bld.
Farnborough, Hants GU14 6TD
UK

Dr H.L.Jones
TASC
1 Jacob Way
Reading, MA 01867
USA

Mr M.C.Muenier
Electronique Serge Dassault
55, Quai Carnot
92214 Saint-Cloud
France

Mr U.Völckers
Institut für Flugführung
DFVLR-Flughafen
3300 Braunschweig
Germany

CONTENTS

OVERVIEW OF AGARD LECTURE SERIES NO. 155

KNOWLEDGE-BASED CONCEPTS AND ARTIFICIAL INTELLIGENCE APPLICATIONS TO GUIDANCE AND CONTROL

Richard Paul Quinlivan
General Electric Company

INTRODUCTION

The use of various forms of Artificial Intelligence (AI) Techniques to help solve problems related to Guidance and Control (G&C) is a subject of current interest in the field. Most, if not all, of the practical applications utilize Knowledge Based techniques to create so-called "Expert Systems." *These systems seem to offer solutions to Guidance and Control problems that have a large judgement content.* Examples include maintenance systems and real-time, decision aiding systems.

The objective of this Lecture Series is to present a number of application oriented lectures augmented by several tutorial lectures, all presented by Guidance and Control practitioners. The lecturers come from several of the participating AGARD countries, specifically the United States, Canada, France, United Kingdom, and West Germany. We have 10 lectures and will conclude with a round table discussion involving all the participants.

GUIDANCE AND CONTROL PERSPECTIVE

The Guidance and Control field has seen, over the past 30 years, the introduction of a very large group of technologies.

In the control system design arena, we have passed from cut and try design techniques based upon frequency domain analysis tools most applicable to single-input, single-output systems through synthesis techniques for multivariable systems involving optimization techniques based in the time domain. We have recently arrived at modern techniques that blend the capability of frequency domain techniques to deal with system modeling uncertainty and the capability of state-space multivariable techniques to deal multiple inputs and outputs to the advantage of both. The formalization of the estimation problem provided by the Kalman filter has clarified the approach to converting measurements into state estimates for controls.

Using these techniques, we have implemented aircraft flight control systems, automatic landing systems, missile guidance systems, automatic navigation systems, and spacecraft guidance and control systems to mention a few.

Sensor technology developments in Radars, Electro-optic Systems, and Inertial Measurement Systems, coupled with advances in communications systems, computers, and displays, have made it possible to gather, process and present to the human operators incredible amounts of information. Unfortunately, not much has been done to assist the operators to act on the information presented.

For example, a modern fighter aircraft cockpit is filled with displays that can present all the data available on the aircraft and often do. Everywhere the hand can reach, and some places it cannot, are switches and buttons to control the various weapons, systems, and displays. The control stick and throttle levers are covered with so many push buttons that a pilot with seven fingers on each hand might have an overwhelming advantage over his more traditionally equipped adversaries. Meanwhile the pilot must keep his eyes outside the cockpit to find, identify, and attack targets and avoid attacks by others all while avoiding ground collision.

The digital computer has impacted G&C from two directions. It makes possible the implementation of large systems utilizing complex algorithms and control logic when imbedded in the vehicle. It also becomes the engineering tool that makes possible the design and analysis of such systems.

The availability of computers that provide ever increasing amounts of computing capability has prompted scientists to grasp at the dream of creating systems which exhibit humanlike intelligence and reasoning powers. This field has been named Artificial Intelligence. *Although true humanlike reasoning capability may be forever beyond our reach, practical capabilities have resulted from this work.*

Knowledge-Based or Expert Systems are the principal topic of this Lecture Series because the theory in this area has reached a level of maturity that makes it possible to construct useful systems which solve Guidance and Control problems.

The systems that will be discussed deal, in some way or other, with problems which normally require human judgment and intervention. These problems have been found to be generally intractable to algorithmic techniques. We will discuss maintenance systems for complex electronic equipment and Expert Systems for air traffic control. We will hear about an Expert System for trajectory management of aircraft, and a thrust toward the unmanned combat aircraft and problems and solutions which that will bring.

For the benefit of those not conversant with AI techniques, including the Director, we will hear several application driven tutorials on the overall technology of AI as applied to the Guidance and Control field, the Knowledge Engineering process, the configuration of an Expert System, and a discussion of issues aimed at real time systems.

The lectures as a whole provide overlap in both the application and the theory areas. I have encouraged this overlap and view it as a strength since it gives the listeners the benefit of several points of view.

OVERVIEWS OF THE LECTURES

Lecture 1:

The first lecture is by Dr. Harold Jones of the United States. It is titled "AI Expert System Technology for Guidance and Control Issues." It is aimed specifically at Expert System issues pertaining to tactical aircraft and deals directly with some of the issues that I raised in the *Introduction*.

A key concept discussed is the distinction between conventional problem-solving techniques and the Expert System approach. The conventional approach produces a deterministic response to all anticipated circumstances but will produce unanticipated responses to unanticipated situations. The Expert approach has additional information built into its Knowledge Base, *approximating the resources* of a skilled problem solver. The Inference engine provides the mechanism to attack the problem with these resources.

Dr. Jones goes on to address Expert System technology issues in the context of applications to combat aircraft.

The first of these deals with real-time AI and requires the system to be capable of keeping up with events, i.e., the problem is changing even as we work toward a solution. In addition, answers are required in a timely manner or they are not relevant.

The next issue is Mission-Critical or Life-Critical Software. This is an important issue in Guidance and Control in more conventional applications, especially flight-critical control systems for manned aircraft. In our attempts to deal with software errors, the software is controlled with extensive documentation, backup software is installed to be triggered upon some predetermined indication of a failure in the primary software, and some design teams have adopted *an approach to redundant systems that requires* dissimilar hardware and *software to eliminate* common faults.

The Expert System software acquires additional capability as time goes on and implicitly has the capability to jump to a conclusion just as a human might. Expert Systems cannot be tested in the same context as conventional systems because of this *opportunistic nature*. The complete software testing that is desirable for mission or flight-critical software is just not possible for an Expert System.

The third issue discussed by Dr. Jones is the interface with all of the conventional information gathering and extracting hardware and software aboard the aircraft. He points out that, for an Expert System to work in the capacity of an advisor to the pilot, it must have all available data.

A most critical technology issue is *the communication* with the pilot. This must take place even as the pilot is in life-threatening circumstances and may be disinclined to talk to his machine. The information passed by the system must be trustworthy even if the Expert System is being ignored. This subject will also be addressed by other lecturers.

The Knowledge Acquisition process for the system is especially complex. It is being gathered for systems with *capabilities never before* tested for a population of users who are distinctly individualistic in how they operate. The Knowledge Base must be gathered in a timely manner to be useful, but there is no way to decide when the process is complete or if all the rules that are included work.

The last of Dr. Jones's technology issues deals with the probable need for concurrent or parallel processing to deal with the very heavy computing load. A partitioned system such as this may be likened to a committee of computers trying to come to a decision when each of them has incomplete data and reasoning power.

Lecture 2:
The second lecture of the series is by Mr. Michel Muenier of France. It is entitled "DEDALE: An Expert System for Analog System Maintenance." Mr. Muenier discusses the need for an Expert System-based maintenance system to deal with today's complex electronic systems. Electronics, especially analog electronics, require *skilled technicians, familiar with the equipment, to successfully diagnose faults*. These people generally are not available at the time and place of need. Automated capabilities are needed to deal with equipment failures in a timely manner.

The Expert System methodology is particularly appropriate because of the separation of the representation of knowledge and its exploitation. This provides the flexibility to allow the knowledge base to be upgraded as the troubleshooting data is acquired.

The main body of the paper is *organized* into four chapters.

The first chapter deals with the knowledge, the current knowledge, and the troubleshooting knowledge.

The second chapter takes us through a *DEDALE* session *from the acquisition of the circuit and malfunction data* through the troubleshooting process implanted in DEDALE.

The third chapter deals with the Knowledge Representation in terms of frames and rules. This provides a detailed discussion of the representation of knowledge within DEDALE. The knowledge is stored in frames as objects, attributes, facets, and values. *The* rules utilize forward chaining and backward chaining and are organized in frames as is the Knowledge Base.

The last chapter deals with the various man-machine interfaces of the system. There are three of these: the expert and the Knowledge Base, general information on the particular circuit under test, and specific information on the circuit under test. The general information is acquired from data files when the circuit is designated to be tested. The specific data is acquired interactively from the technician while he is troubleshooting the circuit.

DEDALE is a working prototype that will be extended within the EMICAT Expert System development environment, which is discussed in Mr. *Muenier's second lecture*.

Lecture 3:
The third lecture is one of two independent lectures dealing with the Air Traffic Control (ATC) problem. It is by Dr. B.A. Bowen of Canada and is titled "An Expert System for Aircraft Conflict Resolution in Dense Airspaces." In this lecture, Dr. Bowen describes the problem faced by Air Traffic Controllers on an everyday basis as one requiring the intelligence to perceive potential conflicts, coupled with the experience to make proper decisions – all in a timely manner. All this must occur under conditions that often are very stressful because of the potential for disaster.

The ATC problem is *characterized* by *a number of rules and procedures* that are decomposable into teachable subtasks. This is done routinely in the teaching of new controllers. Therefore, it fits nicely into the domain of Knowledge-Based systems. Algorithmic techniques have failed to solve the ATC problem because the computational load grows exponentially with the number of aircraft. Portions of the problem are tractable to analytic techniques, therefore Dr. Bowen's system design is a hybrid one.

The prototype system is described in some detail in the paper. The system is first developed for sparse airspace problems, i.e., one-on-one problems, and developed to cover the more realistic problem of dense airspaces where the resolution may cause subsequent conflicts. The subject of convergence is discussed.

Lecture 4:
The fourth lecture is by Dr. Michael Bird and is titled "Application of Knowledge Based Techniques to Aircraft Trajectory and Control." This lecture will discuss the implementation of the Unified Trajectory Control System (UTCS). The UTCS is a hybrid system that uses algorithmic techniques to fulfill the various trajectory generation functions and an Expert System to integrate and select from the various trajectory solutions.

The UTCS utilizes production rules, an Inference Engine, and a system of frames for communicating with the trajectory generation systems. The trajectory generation modules are termed trajectory specialists since each is responsible for a different type of trajectory. The integration of the specialists operations is performed by an Expert System called by Dr. Bird the Trajectory Decision Maker (TDM). The TDM schedules the specialists, makes trade-offs between them, and blends their outputs into the full trajectory solution.

The TDM is based on the use of a production rule system and a frame system. The frame system is the interface to the trajectory specialists.

Dr. Bird discusses in depth the approach to the construction of this hybrid system, the interaction between the TDM and the specialists, and the varying detail required of the trajectories at each stage in the decision process. A concept was developed to deal with the uncertainty level inherent in the trajectory predictions.

UTCS was simulated for a section of a low-altitude combat mission. The simulation included five of the trajectory specialists and flew an aircraft and flight control simulation over a land mass simulation of the Fulda Gap area, including some ground-to-air threats.

The simulation was programmed partially in FORTRAN and partially in LISP.

Lecture 5:
The fifth lecture is by Dr. Brian Ellis and is titled "Towards the Unmanned Cockpit." In his lecture he considers the application of Intelligent Knowledge-Based Systems (IKBS) to the task of replacing the pilot in combat aircraft. The motivation for such a step is the rapidly growing complexity of the pilot's task in the combat environment. Currently, the pilot is required because of his data correlation capability and his ability to jump to intuitive solutions to complex problems. The system that ultimately replaces the pilot will have to exhibit these traits.

According to Dr. Ellis, the path to the unmanned cockpit will begin slowly and carefully with the use of an intelligent assistant which will integrate and provide for the pilot's use the Knowledge Base of multiple Experts. To be useful, the assistant will have to have contextual awareness, examine alternative solutions, and will be self-learning and self-extending. It will be adaptable to the needs of the individual specific pilot and will provide intelligent explanations appropriate to the situation.

Areas that can and will benefit from the application of IKBS include premission and real-time mission planning and a signal processing and data fusion capability to present integrated situation data to the pilot. An intelligent system-monitoring capability to deal with equipment failures will also be valuable. The IKBS will independently control displays so as to present appropriate displays and perform necessary resource allocation.

Dr. Ellis will discuss the present state of IKBS and progress along the path to its ultimate realization.

Lecture 6:
The sixth lecture is by Mr. J. P. Aubert. It is an application lecture dealing with hardware fault diagnostics. The lecture is titled "Toward a General Fault Detection and Maintenance System (The Flag 2 Project)." Mr. Aubert's application is an aircraft navigation system of considerable complexity. The system described is a second-generation Expert System.

The first-generation system consisted of a set of rules and tests, the knowledge of the test costs, the replacement costs, and the failure probabilities of the various components. It operated on the system failures in such a way as to optimize the expected repair costs. Certain criticisms were made of the way in which it worked and are discussed in the lecture. The current system answers these criticisms.

An important change in the second-generation system is the incorporation of a Knowledge Acquisition System (KAS) that allows the Navigation Expert to describe the system in a structural and functional description.

It is clear in Mr. Aubert's lecture that the *incorporation of the Knowledge Acquisition System was the key to the successful implementation* of the Flag 2 Expert System. The KAS is described as general enough to be used in other applications that exhibit similar characteristics to the subject navigation system.

Lecture 7:
The seventh lecture is a tutorial lecture by Dr. Bird titled "A Review of the Knowledge Engineering Process." In this lecture Dr. Bird discusses the concept of the Expert System as a combination of two fundamental parts, the Knowledge Base and the Inference Engine. The Knowledge Base holds the facts, heuristics, and problem-solving rules. The Inference Engine is the procedure for using the Knowledge Base for solving a given problem.

Dr. Bird reviews the various Knowledge Engineering approaches, in particular those that concentrate on using human experts as knowledge sources. The approaches are explained whereby expert knowledge is captured in an appropriate data base. The various techniques for representing knowledge, such as first-order predicate logic, semantic networks, frames, and production rules, are discussed in terms of their structure, advantages, and disadvantages.

His next topic is the knowledge acquisition process, which is subdivided into the identification, conceptualization, formalization, implementation, and testing stages. An important topic in the Knowledge Engineering process is the tools that are required to make the development of Expert Systems practical. In particular, the various AI programming languages such as LISP, PROLOG, OPS5, ART, and KEE are discussed. System-building aids and support facilities are also explained. Dr. Bird closes with a discussion of the validation process in terms of the validation required of normal computer programs.

Lecture 8:
The eighth lecture is titled "A Rule Based System for Arrival Sequencing and Scheduling in Air Traffic Control" and is presented by Dr. Uwe Voelckers. This lecture deals with the Air Traffic Control problem as does Dr. Bowen's first paper.

Dr. Voelckers' lecture opens with an introduction to the Air Traffic Control problem and particularly to the task assigned to the Air Traffic Control Officer (ATCO). The task of the ATCO has remained a manual task even as automation has taken over much of the signal processing. The ACTO still must manually control the aircraft assigned to him within the controlled airspace. A completely automated system, with ACTO monitoring only, is considered unacceptable even if possible. A Knowledge-Based Expert System offers the possibility of providing an acceptable, extendable system that can appropriately advise the ACTO.

He goes on to discuss the current efforts to apply AI techniques to the ATC problem, followed by a brief description of the organization of the Expert System, which is similar to several of the Expert Systems described by other lecturers.

The lecture moves on to cover a more detailed presentation of the arrival sequencing problem as it exists today with a discussion of problems and limitations.

The main topic of the lecture is the discussion of the computer-based arrival planning system with its evolution from the algorithmic-based COMPAS system to the hybrid PLANAIR COMPAS system. The lecture includes a discussion of the architecture, the Inference Element, and the Knowledge Base. The Knowledge Base is subdivided into the static and dynamic ATC knowledge and the rule base. The planning strategies are discussed in some detail.

Dr. Voelckers contends that the rule-based PLANAIR system establishes a proper planning sequence according to rules that are used by ATCOs.

Lecture 9:
The ninth lecture is by Mr. Muenier and is one of our tutorial lectures. It is titled "How to Use PROLOG for Expert System Development." In his lecture Mr. Muenier discusses the use of PROLOG as a language for Expert System development and the necessary environment and additional facilities for industrial use.

PROLOG has been favored for Expert System development by some groups but has been disappointing in other efforts. This contradiction arises because several approaches are possible when PROLOG is used as a development tool. One approach is to use PROLOG directly as a specification language and use the direct features of the language as a reasoning mechanism. Another approach is to use PROLOG as the implementation language. In this case, the knowledge formalism is defined as required and implemented in PROLOG. Both approaches have deficiencies that Mr. Muenier discusses.

The remainder of the paper discusses EMICAT (M14), which is an environment developed by Mr. Muenier and his associates at Electronique Serge DASSAULT specifically as an industrial environment for the development of Expert Systems.

EMICAT integrates the advantages of the PROLOG language with the additional features and capabilities required to fulfill the design team's needs. These include advanced Expert Systems features as well as services such as graphics support, editing capability, and Knowledge-Base archiving required in the development of real products.

Lecture 10:
The tenth and last lecture is by Dr. Bowen and is titled "Real Time Expert Systems: A Status Report." In this lecture Dr. Bowen discusses and reviews the current state of the art with respect to real-time applications of Expert Systems. Many of the Guidance and Control applications of Expert Systems discussed during this lecture series fit into this category so it is a subject of much interest.

Dr. Bowen gives us a perspective on the control problem in general and goes on to describe some reasons why an Expert System could be the preferred approach to a system solution.

In keeping with the title, he discusses the nature of the real-time problem and exposes the system requirements that flow from the real-time consideration.

Dr. Bowen goes on to review and describe 10 applications of Knowledge-Based techniques to control system examples. These examples range from military to process control and are programmed in LISP, Pascal, OPS5, and PROLOG. In his reviews he covers problem domain, design goals, architecture, knowledge representation, inferencing, language, status, performance assessment, and future work.

At the conclusion of the system reviews, Dr. Bowen gives us his view of unresolved issues and problems. These include the dominance of surface-level considerations to the exclusion of deep knowledge of the process physics. The systems deal only with static sets of system parameters and cannot deal with dynamic problems. Additional issues discussed include Knowledge-Base representation and consistency.

CONCLUSION

It is clear that this lecture series has brought together a representative number of applications of Knowledge-Based systems that spans the field of Guidance and Control. The difficult issues in the various application areas have been uncovered and illustrated

REFERENCES

1. Jones, H., "AI Expert System Technology for Guidance and Control Issues," AGARD Lecture Series No. 155

2. Muenier, M., "DEDALE: An Expert System for Analog System Maintenance," AGARD Lecture Series No. 155

3. Bowen, B. A., "An Expert System for Aircraft Conflict Resolution in Dense Airspaces," AGARD Lecture Series No. 155

4. Bird, M., "Application of Knowledge Based Techniques to Aircraft Trajectory And Control," AGARD Lecture Series No. 155

5. Ellis, B., "Towards the Unmanned Cockpit," AGARD Lecture Series No. 155

6. Aubert, J. P., "Toward a General Fault Detection and Maintenance System (The Flac 2 Project)," AGARD Lecture Series No. 155.

7. Bird, M., "A Review of the Knowledge Engineering Process," AGARD Lecture Series No. 155

8. Voelckers, U., "A Rule Based System for Arrival Sequencing," AGARD Lecture Series No. 155

9. Muenier, M., "How to Use PROLOG for Expert System Development," AGARD Lecture Series No. 155

10. Bowen, B. A., "Real Time Expert Systems: A Status Report," AGARD Lecture Series No. 155

AI EXPERT SYSTEM TECHNOLOGY ISSUES FOR GUIDANCE AND CONTROL APPLICATIONS

by

Harold L. Jones
The Analytic Sciences Corporation
1 Jacob Way
Reading, MA 08167, USA

ABSTRACT

The increasing sophistication of modern digital avionics could conceivably overload the system management capabilities of the tactical aircraft pilot. Indeed, field interviews have verified that the operational pilot's role includes significant time in managing complex electronics systems. Often the pilot must decide which systems to trust both mission success and personal survival to with only a rudimentary comprehension of the system components and their behavior. The AI field of expert systems could make a substantial contribution toward improving both aircraft mission effectiveness and the pilot's sense of situation awareness. To meet this challenge, however, the research and development community must resolve a number of signficant technical issues which could otherwise limit the capability and acceptability of expert systems for coping with mission-critical flight situations. This paper provides a perspective on a set of technical issues which, if unresolved, could limit the capability and acceptability of expert systems decisionmaking for avionics applications. Examples from on-going expert system development programs are used to illustrate likely architectures and applications of future intelligent avionic systems.

INTRODUCTION

The advent of powerful, relatively economical LISP machines in the past few years has generated intense interest in expert systems applications within the United States' research and development community. There have been a handful of acknowledged successes, and an abundance of new, promising applications in a wide range of disciplines. For the most part, these expert systems perform diagnostic or tutorial functions, typically in an environment which permits direct human oversight and which imposes only loose requirements for timely decisionmaking.

Recent initiatives within the Department of Defense have begun to focus attention on the potential of expert systems for alleviating the pilot sensory overload problem in next-generation tactical aircraft, and for adaptively tailoring the aircraft performance characteristics to specific mission objectives and circumstances. In doing so, the DoD has selected an application domain characterized by the need for mission- and life-critical decisionmaking under rigid time constraints. Furthermore, an intelligent avionics system must perform this decisionmaking based on imprecise and possibly incomplete knowledge of the pilot's perception of the environment, his assessment of the viable mission options, or his ability to execute his mission choices. The demands of this applications domain pose new challenges to both the avionics and AI communities. This paper provides a perspective on the underlying technical issues and uses examples from ongoing intelligent avionics systems programs.

EXPERT SYSTEM PERSPECTIVE

Artificial intelligence applications can be partitioned into four related disciplines — natural language processing, image understanding, robotics, and expert systems. Each of these has a possible role in an intelligent avionics system, and all are being pursued within the United States' R&D community. A natural language interface between the pilot and the aircraft avionics system would improve and simplify the flow of information within the cockpit. Similarly, an image understanding [illegible] support robust automatic target recognition (ATR) for ground strike missions using digital targeting sensor imagery (radar, infrared, or electro-optical). A more generalized out-the-window vision system (possibly using radar or infrared sensors) would be [illegible] if robotics are to see significant applications in an avionics system.

By far the broadest area of pursuit is the application of expert systems technology to advanced avionics, with symposia such as NAECON devoting multiple sessions to the topic. One example is the Expert Navigator (Ref. 1), which was the source from which many of the perspectives presented in this paper were derived. Although the technology discussions presented in this paper are pertinent to the other AI disciplines, the discussion is specifically focused on the application of expert system technology to advanced avionics.

The recent preoccupation with LISP machines and rule-based (If-Then) expert systems has had the unfortunate effect of blurring the distinction between AI and conventional problem solving paradigms. It is not the If-Then structures which set expert systems apart from conventional, numerically-driven computer programs, but rather the [illegible] which knowledge about the problem is represented and solutions are pursued.

In the conventional problem solving paradigm, the designer uses his judgement and skill to select a means to solve a problem, and then writes numerically-driven procedures which implement his selected approach (Fig. 1). Conditional logic may be used to select optimal parameter values or to test for applicability of a given procedure, but in essence the designer of conventional software must anticipate all possible circumstances in which the system might be used and provide a set of procedural solutions applicable across that range.

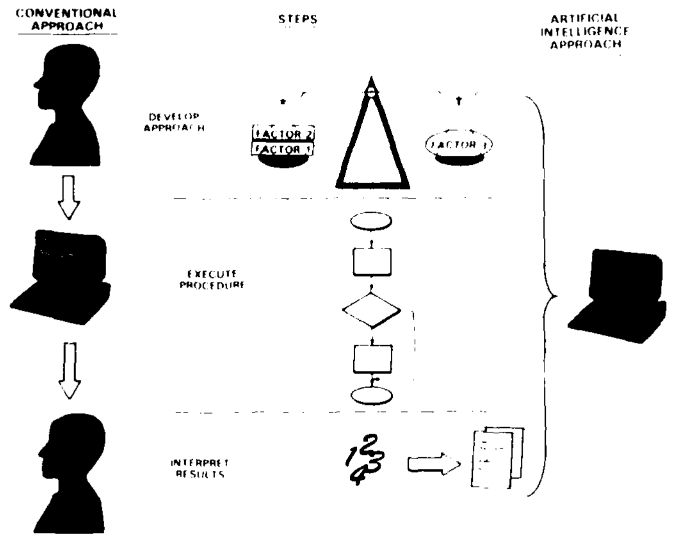


Figure 1 Comparison of Problem Solving Paradigms

By contrast, AI addresses the problem solving process at a more fundamental level, focusing on the underlying constraints, relationships and goals which define an acceptable solution and dictate the manner in which it must be formulated. Thus, an expert system seeks to emulate the planning and problem solving skill of the designer. As illustrated in Fig. 2, the knowledge base on which the expert system reasons encompasses the system and mission constraints, performance and mission goals, heuristic rules, value judgements, and procedures parts (e.g., algorithms) used by a skilled problem solver. The inference engine constitutes a focusing mechanism for applying the knowledge base to the system data in order to postulate and examine alternative courses of action, and select the course best suited to attaining the stated goals. The intended result is a problem solving capability which is extremely robust in responding to unanticipated circumstances and operating conditions.

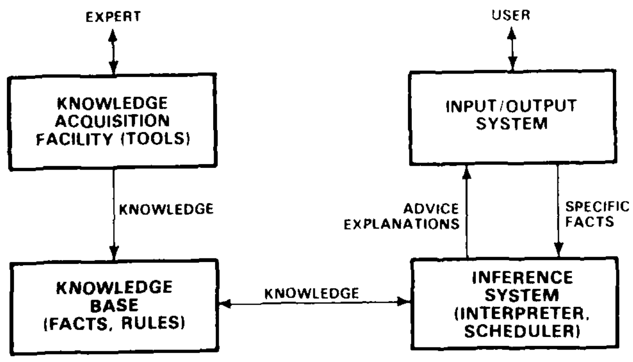

EXPERT
USER
KNOWLEDGE ACQUISITION FACILITY (TOOLS)
INPUT/OUTPUT SYSTEM
KNOWLEDGE
ADVICE EXPLANATIONS
SPECIFIC FACTS
KNOWLEDGE BASE (FACTS, RULES)
KNOWLEDGE
INFERENCE SYSTEM (INTERPRETER, SCHEDULER)

Figure 2 Basic Expert Systems Architecture

An expert system is not likely to follow a rigid "recipe" in formulating a solution. Rather, it is often opportunistic, using incomplete data to posit plausible, but as yet unproven, solutions and then accumulating evidence in favor of (or against) the strawman. The problem of efficiently focusing an expert system on the most productive line of reasoning is a largely unattained research objective, with the most effective search strategy generally being either problem- or scenario-specific. Regardless of the search strategy pursued, however, one of the fundamental traits of an expert system is the ability to explain its reasoning by summarizing the evidence supporting a given conclusion.

Finally, although If-Then rules are the most commonly used knowledge representation form, other representations may be more appropriate to a given application. A rule-based system is very easy to augment with additional rules and is well adapted to applications involving sequences of operations, but a large rule set can be difficult to test for completeness (Are all the necessary rules there?) and can result in relatively slow operation of the expert system. For these reasons, frames and semantic nets (Fig. 3) are clearly preferable where their imposed semantics are appropriate to the problem.

EXPERT SYSTEM TECHNOLOGY ISSUES

Development of an intelligent avionics suite poses three largely unaddressed problem areas for the AI research community. First, the expert system would be given at least partial responsibility for mission- or life-critical decisions, and thus must be virtually infallible, even in unanticipated contingency circumstances. Second, an intelligent avionics system would be expected to function within rigid time constraints imposed by mission and aircraft operations, and applications such as flight control or threat avoidance could require response in fractions of a second. And, finally, an intelligent avionics system's ultimate goal will be to maximize the pilot's offensive and defensive fighting capability; however, the pilot's situation awareness and performance level will be extremely difficult to assess. Thus, the expert system will be required to function with an incomplete, and possibly erroneous, set of priorities and performance goals.

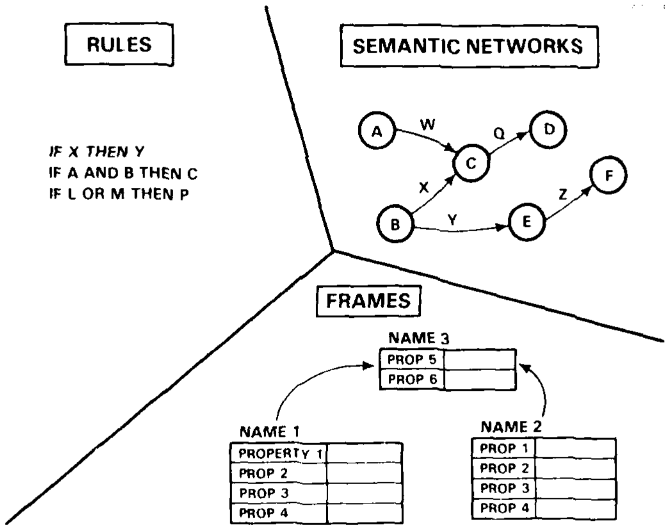


Figure 3 Knowledge Representation Forms

These three new problem areas can be translated into six basic technology issues, as synopsized in Fig. 4. The relevance of each issue to the application of AI to avionics problems is discussed in the remainder of this section.

Real-time AI - In the traditional problem solving paradigm, an expert system accepts status information from the external world, and then searches for all possible explanations in order to find the best solution or response. Considering threat evasion as an example, the planning action required is to search for a sequence of actions that will lead to the desired resultant state. The problem is that while this planning is taking place, the state of the system is changing, so that the arrived-at plan may no longer be applicable. Similar examples could be drawn from flight control or targeting applications.

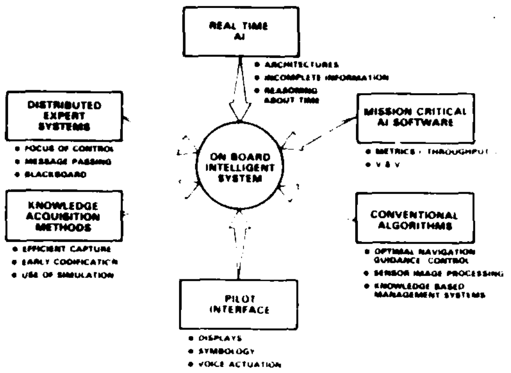


Figure 4   Key Technologies of Intelligent Avionics Systems

The problem statement for an intelligent avionics system forces the expert system to reason about the available time (or simply accept an imposed time constraint) and then obtain the "best" solution consistent with the available time, computer, and pilot interface resources. It must anticipate the future state of the system based on the current observations, and recognize that currently valid hypotheses (e.g., the aircraft can evade a threat SAM radar prior to missile firing) may become invalid at some future time, as would all subsequent conclusions predicated on such hypotheses being true. This problem is referred to as "truth maintenance". Recent releases of development tools (e.g., KEE™ and ART™) provide a truth maintenance capability; however, the theory behind truth maintenance is an active research area (Ref. 9).

One obvious consequence of real-time operation is the necessity of reasoning with partial or incomplete information. In addition to the obvious facts that key event may be unobservable (has a SAM been fired?) or not relayed by the pilot, there may not be sufficient time to process the information which is available. This requires an ability to focus resources on important goals and promising lines of reasoning, without fully searching all possible actions and consequences.

Techniques for reasoning with incomplete information generally employ a priori assignment of weighting fact rs (probabilities, levels of belief, confidence factors) for the antecedent components of a rule and use incoming observations (evidence) along with prescribed rules of combination (e.g., Bayes' rule) to incrementally update the likelihood of active hypotheses (Refs. 10 through 12). The evidence accrual techniques being pursued by Glasson (Ref. 2) and Green (Ref. 5) use estimates of the incremental evidence attainable along alternative search paths as a mechanism for truncating low-payoff search sequences.

Mission-critical Software - The knowledge base in an expert system isn't static. As new facts are learned through the inference process or through trial-and-error interactions with the external world, the knowledge contained in the system expands. Similarly, the inferencing sequence pursued by an opportunistic expert system is not a priori predictable, or necessarily repeatable, for a given set of initial conditions. The code can be checked for functional correctness, and the expert system can be exercised to obtain statistical measures of completeness of the knowledge base, correctness of the resulting solution, and elapsed time for reaching a decision. However, a complete or statistically optimum set of test paths cannot be defined in a classical sense. Thus, the ultimate measure of software quality is likely to be "trustworthiness" rather than "correctness."

Conventional Algorithms - As previously mentioned, the inputs provided by the pilot are likely to provide an avionics expert system with an incomplete (or uncertain) perspective on the external environment and the pilot/aircraft system status. Furthermore, the pilot's attention may be drawn to more pressing matters, making his implementation of recommended actions unpredictable.

A partial resolution of this dilemma lies in effective interrogative and control interfaces between the avionics expert system and the various avionics subsystems. The procedure set which embodies the expert system's external world interface should incorporate state-of-the-art statistical tests and signal processing algorithms, including

™KEE is a trademark of Intellicorp.

™ART is a trademark of Inference Corporation.

model-based fault diagnosis tests, optimal navigation filters and control algorithms, optimal route planners, and model-based target recognition algorithms for exploiting EW and targeting sensor outputs. In essence, to be effective, an avionics expert system must have at its disposal the best possible conventional information extraction and planning techniques.

Pilot Interface - The traditional avionics concern addresses the mechanics of the pilot interface -- the display technology and symbology to be used, the viability of voice recognition in a high stress environment, etc. Effective pilot/avionics suite communications is even more important for an intelligent avionics system, however, because of the need for two-way information exchange. The pilot is the ultimate source of mission priorities, and a vital source of status and situation awareness information. As a consequence, pilot attention becomes an important resource for an expert system to manage.

The expert system must be able to ensure that high priority messages (e.g., recommended evasive maneuvers) are understood by the pilot, and must be capable of providing a broad range of system status and inference explanatory information on an as-needed basis. It must be able to attach the appropriate priority and interpretation to information volunteered by the pilot, but must also be able to achieve reasonable system goals in the absence of pilot data inputs.

Mission success is a joint pilot/avionics system responsibility, with the pilot having ultimate authority. In essence, an intelligent avionics system must exert a level of decisionmaking autonomy appropriate to the circumstances. One means of accomplishing this would be to estimate pilot workload and then adapt data exchange and autonomy levels accordingly. Unfortunately, there are no reliable metrics for in situ assessment of pilot workload or attention level. In the absence of such metrics, an intelligent avionics system would have to resort to pilot-programmable communication prioritization, which would be accomplished during ground-based mission planning.

Knowledge Acquisition - Formal operational training for tactical pilots stresses reliance on simple fundamentals for mission execution. As an example, pilots are schooled heavily in navigation using only a compass and chronometer, even though most will have access to a variety of sophisticated aided-inertial navigation systems during their careers. The result is a widely varied experience base among operational pilots, with ad hoc rules and procedures based upon personal experience forming an important component of each pilot's knowledge base. This user-to-user variability invalidates the concept of a single "correct" knowledge base in lieu of a broader knowledge base which can be tailored to the specific needs of individual pilots. The knowledge acquisition problem is particularly acute in planning for future avionics systems because of the absence of operational domain expertise with developmental avionics systems, some of which have not even been flight tested.

Much attention has been focused on tools for acquisition and codification of knowledge -- KEE and ART are representive of the current state-of-the-art. Nonetheless, extraction of relevant knowledge from domain experts in an architecture-independent form represents a formidable task. Procedural techniques, such as partitioning the knowledge into "chunks" along functional or domain expertise lines, can facilitate the process; however, there are no viable methodologies for assessing the completeness of the resulting data base, or whether or not the functionality is appropriate to the problem. Similar to the problem of software validation, assessing the quality of the knowledge base is largely a trial-and-error process.

Distributed Expert Systems - The new supercomputing architectures stress parallel computing, a trend which is also present in most advanced avionics architectures. Because of the potentially substantial computational load associated with most nontrivial expert system designs, and because of the need to modularize the system to facilitate development and testing, an avionics expert system is likely to require distributed (parallel) processing.

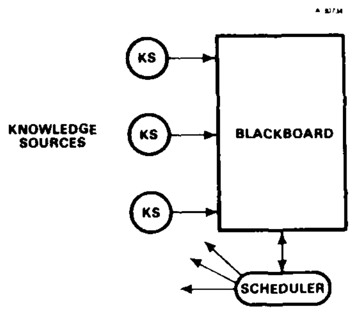
In developing an expert system architecture for a distributed computing environment, the principal issues of interest are:

- Distributability - Can the computation and inferencing be distributed to the available processors without imposing unreasonable time resource constraints or excessive control overhead.
- Concurrency - Can simultaneous hypothesis formulation and testing be efficiently implemented, with minimal extraneous searching and acceptable control overhead.
- Focus of control - Can the inferencing process be effectively focused on those lines of reasoning most likely to lead to a viable solution.
- Real-time operation - Will the architecture support timely decisions based on incomplete data.

- Modularity - Will the proposed system modularization support efficient design, knowledge acquisition, software development, and testing.

Current applications programs are focused on two basic architectural options for meeting the stated requirements: blackboard architectures and communicating architectures (Fig. 5). Both employ modular expert systems (knowledge sources and expert objects, respectively), each module reasoning only within a specified domain. The primary differences are in the global control philosophy.

Blackboard



Communicating Expert Objects

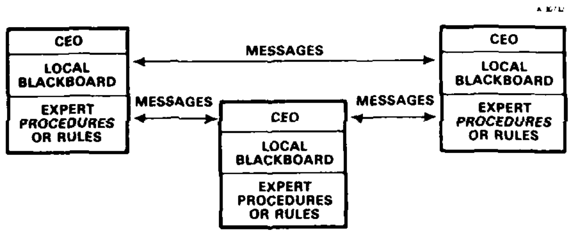


Figure 5 Candidate Architectures for Distributed Expert Systems

A blackboard system uses a scheduler to select the appropriate sequence of knowledge source invocations, but the scheduler can be both inflexible and slow. Communicating architectures provide much more autonomy for the expert objects and, as a consequence, can be much more opportunistic in pursuing promising lines of reasoning. However, weak evidence can potentially lead to high message volumes and inconclusive searches. Both architectures have achieved impressive successes (Refs. 4 through 8), but both require considerable skill to effectively implement.

SUMMARY

The function of an intelligent avionics system should be to perform pilot-delegated, mission-critical functions. Since the pilot must bear ultimate responsibility for mission success and safety, the role of an expert system would be to recommend intelligent options for system management and mission planning, and to exercise the level of autonomy conferred by the pilot in implementing those options. In performing this role, the expert system could use a combination of heuristics (pilot goals and beliefs) and mathematical procedures (signal processing algorithms, decision models and strategy generators, etc.) in responding to unforeseen contingencies.

It seems readily apparent that intelligent avionics have the potential for improving aircraft performance and mission-responsiveness, while decreasing pilot workload. Because avionics perform a real-time, mission-critical function, however, full exploitation of the promise of AI will require substantial advances in a number of critical technology areas. The DoD is currently pursuing a number of program initiatives with the objective of demonstrating the applicability of AI to the avionics problem. These initiatives will contribute substantially to the state-of-the-art in AI applications; however, solutions to several of the pertinent technology issues are likely to require contributions from the broader AI research community.

ACKNOWLEDGEMENT

The author would like to thank Mr. Douglas P. Glasson of The Analytic Sciences Corporation and Professor Peter E. Green of Worchester Polytechnic Institute for many of the ideas and perspectives presented in this paper.

REFERENCES

1. Jones, H.L., et al, "An Expert Systems Approach to Adaptive Tactical Navigation," Proceedings of the First Conference on Artificial Intelligence Applications, Denver, CO, December 1984.

2. Glasson, D.P, and Pomarede, J.L., "Adaptive Tactical Navigation - Phase II Concept Development," The Analytic Sciences Corporation, Technical Report AFWAL-TR-86-1066.

3. Glasson, D.P., and Matchett, G.A., "Recommended Basic Navigation and Expert Navigator Architectures," The Analytic Sciences Corporation, Technical Information Memorandum TIM-5344-2, November 1986.

4. Nii, H.P., "Blackboard Systems: The Blackboard Model of Problem Solving and the Evolution of Blackboard Architectures," The AI Magazine, Summer 1986, pp. 38-53.

5. Green, P.E., "Resource Limitation Issues in Real-Time Intelligent Systems," 1986 SPIE Application of Artificial Intelligence Conference, Orlando, FL, April 1986.

6. Lesser, V.R., and Erman, L.D., "An Experiment in Distributed Interpretation," IEEE Trans. on Computers, Vol. 29, No. 12, pp. 1144-1163, December 1980.

7. Winston, P.H., Artificial Intelligence, Addison-Wesley Publishing Co., Reading, MA, 1984.

8. Wyss, S.M., "A Software Methodology for Distributed Real-Time Intelligent Systems," Worcester Polytechnic Institute, March 1986.

9. de Kleer, J., "An Assumption-based TMS," Artificial Intelligence, Vol. 28, 1986.

10. Shafer, G., "A Mathematical Theory of Evidence," Princeton University Press, Princeton, 1976.

11. Negoita, C.V., "Expert Systems and Fuzzy Systems," Benjamin/Cummings, Menlo Park, CA, 1985.

12. Prade, H. "A Computational Approach to Approximate and Plausible Reasoning with Applications to Expert Systems," IEEE Transactions on Pattern Analysis and Machine Intelligence (3), pp. 260-283, May 1985.

DEDALE :

AN EXPERT SYSTEM FOR ANALOG CIRCUIT MAINTENANCE

Michel MESNIER - Philippe DEVES

ELECTRONIQUE SERGE DASSAULT

92214 SAINT-CLOUD - FRANCE

INTRODUCTION

If technological progress is very rapidly increasing the capacity of electronic circuits, on the other hand the difficulty of establishing diagnoses in the event of malfunction is increasing even more quickly.

Repair time bears directly on the availability of an equipment and its cost of maintenance. A repair operation may be divided into the following three distinct phases :

- Test : detection of the fault condition
- Diagnosis : localization of the fault
- Repair : return to a normal serviceable condition.

The first and third of these phases are amenable to systematic procedures. The second phase, however, requires not only in-depth knowledge of the equipment to be repaired but also of its application, since a defective part may induce malfunction of other elements : how can one be sure of the serviceability of an engine starter if the car battery is low ? This simple example points to the difficulty of diagnostics in the case of a highly complex electronic circuit in which interaction is at a very high level.

It is necessary to distinguish two broad types of circuit :

a) Digital circuits which are amenable to troubleshooting algorithms : automatic test equipment has been used by the industry for many years for this type of circuit.

b) Analog circuits which are very difficult to describe, especially outside of their normal ranges of operation. At the present time, troubleshooting is performed by highly qualified technicians. This dependence on such specialists, little motivated by this kind of work, is not without problem. Such persons are not numerous and therefore cannot be close to all equipment in service. Moreover, their mission ends rather quickly in the life of equipment in operation, since they are then assigned to a new equipment.

Automation, i.e. computer processing, of analog circuit troubleshooting is thus a crucial requirement.

Conventional data processing does not offer a satisfactory response to this requirement. In particular, it comes up against the difficulty of quantifying information to be processed and also against the continuum of fault conditions. It is necessary to be capable of reasoning on a fault often appearing for the first time. Artificial intelligence attempts to provide this ability of adaptation to unforeseeable situations, otherwise a major weakness of conventional data processing.

Among artificial intelligence techniques, those of expert systems (E.S.) would appear to be those immediately applicable in an industrial environment. A particularly convenient feature of E.S. is independence between the representation and exploitation of knowledge. Another interesting characteristic of E.S. due to their declarative programming, is their ease of modification. They apply indeed to areas where knowledge is poorly formalized and therefore never completely analyzed at the start of a project. In particular, experience is handed down only progressively through the analysis of system failures.

DEDALE is an E.S. developed by Electronique Serge DASSAULT, the objective of which is to cover the diagnostic phase of an analog circuit.

DEDALE allows the fast and intelligent identification of circuit faults. Its ability is not limited to a few circuits ; it is able to troubleshoot fairly quickly any new circuit with minimum initial knowledge. The novelty of DEDALE compared with other E.S. largely results from this objective.

The procedure adopted by DEDALE in producing diagnoses is comparable with that of an expert faced with the same situation. Both apply very general knowledge born of experience (expertise) to a situation they discover, one by means of a structural and functional description of the defective circuit and the other by means of a circuit diagram and layout drawing, a parts list and information resulting from electrical tests. The system, no more than the expert, is incapable of identifying a fault without additional information. A troubleshooter is able to use test and stimulation equipment in addition to his five senses. DEDALE must therefore make use of the troubleshooter's competence for completing or supplementing its information. This constraint, which manifests itself by dialoguing throughout the repair work, has heavy consequences. It implies an effort to limit the number of information requests and to monitor their pertinence.

DEDALE has been designed in collaboration with the Scientific Centre of IBM FRANCE. Initial meetings between technicians took place in 1983 and were officialized by a research agreement signed at the beginning of 1985.

The study resulted in the production of an experimental model which was demonstrated in the course of 1985.

CHAPTER I - THE KNOWLEDGE

The computer modellization and representation of knowledge useful for repairing analog electronic circuits appeared to be the most crucial and complex phases of DEDALE development. It would appear to be this which characterizes software of the E.S. type compared with more conventional software for which the representation of information raises in general few problems.

1.1 CIRCUIT KNOWLEDGE

This knowledge relates to the structural and functional aspects of a circuit on which diagnostics are based. [1] [3]

Basic electronic concepts, general for all circuits, enable a particular circuit to be characterized and referenced. Their internal structures are adapted to effective use during diagnostics. The present method adopted for organizing and controlling this information ("frames") highlights the hierarchical and relational aspect of a fact within a context. Implementation using object oriented languages is perfectly conceivable. Each circuit is the subject of a dual description : structural and functional.

1.1.1 Structural Description

The structural description attempts to reproduce in DEDALE all information useful for troubleshooting obtained by visual observation of the circuit or its layout drawing.

It is used for refining a diagnosis by use of spatial concepts (location, dimensions, proximity) or technological concepts (nature of connections, components) and also for identifying physical locations at which tests must be performed. This identification, which appears to the user of DEDALE in the form of a graphical view, is a first step towards complete automation of the process.

The following various structural concepts have been adopted :

- structural_block	: a structural_block is a physical entity identified by its location and dimensions, by its technology, by test points and by an internal structural breakdown.
- atom	: this is a structural_block for which further structural breakdown is of no additional interest for diagnostics. Consequently, it is the smallest entity which can be suspected in a case of malfunction. An atom must possess electronic behaviour known to DEDALE, i.e. must correspond to a function (refer to § 1.1.2).
- internal_node	: this is a physical entity combining technologically homogeneous connections within a structural block.
- link	: this is an atom having the special function of electrical connection. The assumption of normal operation of the electronical nodes of a circuit is in general quickly admitted in the course of troubleshooting, since it enables a functional fault to be identified very rapidly. It is for this reason that links and internal_nodes are distinguished from atoms and structural_blocks.
- test_point	: physical location at which an observation of any nature may be envisaged. Example : measurement of a voltage. The test_point is generally located in an internal_node of the circuit.

Structural blocks and internal nodes enable the circuit to be described in the form of a hierarchy of structural entities. The circuit represents the structural_block at the highest level, while the atom is that of the lowest level. The characteristics of each entity are defined in a spatial reference frame.

The size of a block is given by the length and width of the rectangle in which the block is written.

1.1.2 Functional Description

The functional description of a circuit reproduces in DEDALE a conventional functional analysis of the circuit.

This analysis must relate to troubleshooting and not to the functional design of the circuit. It is for this reason that it must involve electronic functions to which DEDALE can refer during diagnosis, as well as functional breakdowns enabling the problem to be quickly reduced to a few functions.

The basic functional concepts allowing this description are :

- function : identified by a specific function name and a set of parameters.
  Example : R1 Resistor (N1, N2, 3K) indicates that resistor R1 links nodes N1 and N2 [illegible] value of 3 Kohms. It can also possess several functional breakdowns.
  Example : R1 which may be broken down into :
  R1 Resistor (N1, N3, [illegible])
  and R2 Resistor (N3, N2, 2K). In most cases, this function is predefined, the parameters then have a given semantic specific to this function. Otherwise the parameters are [illegible] its nodes and the function must be broken down.

- node : functional entity assembling equipotential electrical parameters. It enables [illegible] defined as parameters of a function and enables the electrical links existing between the [illegible] functions to be established.

Functions and nodes enable the circuit to be described [illegible] entities, the atom (see § [illegible]) constituting the functional entity at the lowest level, and [illegible] the link with structural decomposition.

3.2 TROUBLESHOOTING KNOWLEDGE

Troubleshooting knowledge is [illegible] given circuit. The purpose of this knowledge is to lead troubleshooting [illegible] using in a pertinent manner the information derived from the circuit description [illegible] measurements, etc.) [2, 3].

This knowledge is represented by inferential rules [illegible] if_[illegible] attached with frames [illegible] programs in Prolog language [illegible] four categories :

. rules of experience

. rules of operation,

. rules of malfunction,

. rules of selection.

3.2.1 Rules of Experience

This type of knowledge based on experience allows [illegible] functions of the circuit from acquired observations. In general, these are suspicions to be confirmed [illegible] faults observed.

Example : If distortion is observed [illegible] amplification, the circuit [illegible] is to be suspected.

These rules are examined one after another in order to determine all possible hypotheses [illegible] are said to be activated in a forward chaining mode.

3.2.2 Rules of Operation

Knowledge of the normal operation of an electronic function [illegible] to be [illegible] or rejected, thereby altering the course of the diagnosis. These rules are [illegible] for troubleshooting. In no way do they constitute rigorous functional test means and consequently [illegible] provide formal proof of normal or abnormal operation.

Example : If the voltage between the anode and cathode of a diode lies between [illegible] and if there is a nonzero positive current from anode to cathode, the diode is operating correctly.

These rules are examined with a precise goal : to demonstrate that a function is operating normally. The related data processing technique is called backward chaining.

During the evaluation of a rule, certain observations may be requested or deduced from other observations (example : voltages). They are obtained by means of the "if_needed" attachment of frames which, when an information is not yet known to DEDALE, activate a procedure to find this information (such as a question asked to the user or the execution of a measurement).

3.2.3 Rules of Malfunction

The principles and objectives of this type of knowledge are similar to those of the rules of operation, except that they attempt to confirm opposite hypotheses. Failure to confirm a hypothesis of normal operation does not indicate the causes of failure. Rules of malfunction can then rapidly guide diagnosis by indicating these causes which may be later used by the rules of experience.

It is also possible to use the rules of malfunction directly if [illegible]potheses concerning malfunction are provided.

Example : If the actual measured frequency of an oscillator is a multiple of its normal frequency, the oscillator is defective and presents the problem of being locked onto a harmonic.

1.2.4 Rules of Selection

The rules of selection (meta-rules) participate in formulating a strategy for troubleshooting a circuit. This knowledge is thus located at a less specific level of the expertise domain (example : to privilege the validation of a hypothesis relating to a subfunction of a function confirmed as defective).

The rules of selection allow faster progression in the process of diagnosis by giving preference to hypotheses likely to prove more fruitful.

1.2.5 Use of Knowledge of the Inferential Type

The rules of operation and malfunction derive from interpretation of the laws of electronics. They are easily acquired and little subject to modification other than their enriching.

The rules of experience correspond to the synthesis of situations experienced by repairmen. They are more difficult to acquire and are never exhaustive.

The rules of selection describe methods of diagnosis. They vary from one expert to another and may be contradictory. They must therefore always be challenged during development.

In addition, DEDALE uses a mode of non-monotonic reasoning, i.e. a rule may be challenged following its application.

1.2.6 Specific Programs

There is certain knowledge relating to circuit troubleshooting, the nature and use of which allow a very conventional data-processing description of the algorithm type.

a) Analysis of Electrical Interaction Between Functions

The analysis of electrical interactions is based on the following simple idea : one function in general influences the operation of another function if there is a possible electrical path between them (it being possible to envisage other cases, such as thermo-electric, inductive and mechanical effects, etc).

This analysis is used above all by the rules of selection in order to limit the area of search.

Example : If a function of the circuit is not performed, then consider only those hypotheses relating to functions affecting the former.

b) Analysis of Priorities

A degree of priority expresses the doubt generally assigned to a structural element, block or atom concerning its electronic performance. Doubt acquired by experience with regard to a structural element does not take into account any special circumstances having produced it. This knowledge may therefore be used whatever the troubleshooting in progress.

The analysis of priorities and the analysis of interactions provide the input data for the rules of selection.

c) Electrical Parameters

When they are not measured directly, electrical parameters may be calculated or deduced. Knowledge of them is essential for determining whether or not an electronic function is normal.

Electrical voltage is an electrical parameter associated with the node entity considered as operating normally. A voltage measurement made on a physical point of a node is sufficient for determining the voltage of this node. This may also be determined by applying Kirchoff's Law relating to voltages. DEDALE attempts to apply this law systematically to avoid making unnecessary measurements.

The parameter of electrical current is a special case, since it is a practically unmeasurable quantity possessing rather the characteristics of a quantitative synthesis of considerations relating to the operation and interaction of functions.

The solution to this problem constitutes part of future DEDALE development. At the present time, DEDALE assumes it is possible to measure current and to apply Kirchoff's Law relating to currents. A program determines the value of an output current for a given function and node from the values of output currents for other functions and for the node considered.

CHAPTER 2 : A DEDALE SESSION

A session of circuit troubleshooting by DEDALE is executed as follows :

2.1 ACQUISITION OF DATA DESCRIBING THE CIRCUIT TO BE TESTED

These data, both structural and functional, are provided in a precise syntax reflecting their hierarchical character. Special care has been taken to facilitate future extensions of this syntax.

A Prolog program immediately translates these data by creating instances of prototype frames relating to the description of a general circuit. Following this phase, the system thus possesses the structural and functional data of the circuit to be tested in a form directly usable by the system.

2.2 ACQUISITION OF QUALITATIVE DATA CONCERNING MALFUNCTION

These are relating to the defective nature of the circuit as shown by the automatic test procedures as follows :

- The list of output signals which differ from those expected in the case of normal operation.

  The system then performs an interaction analysis to determine the higher-level functions to be suspected according to the interaction they have on the circuit outputs.

- If possible, the types of qualitative fault observed on these outputs (signal absent, incorrect frequency, low voltage, etc.) or at the level of the circuit itself (excessive power consumption, etc.). These data are used by the rules of experience.

At the present time, these data are enterred via the console by the operator in reply to questions asked by the system. They could be entered directly by means of an interface between the system and test procedures. This is also the case for all data to be supplied later (mainly the results of voltage measurement), the acquisition of which will be automated in the future.

Given these preliminary data for guiding its diagnosis (in the absence of such data the system operates blindly by investigating all functions), the system starts the actual reasoning (troubleshooting) phase.

2.3 TROUBLESHOOTING

The initiation of troubleshooting is the only procedural part (2 lines of code !) of DEDALE. Troubleshooting itself is performed according to the following cycle.

2.3.1 Application of Rules of Experience

All the rules of experience are examined in the forward chaining mode until saturation of the current fact base. They determine new possible hypotheses (including in particular in-depth exploration of a suspected function) resulting from the fault symptoms observed or deduced by the system in the course of former cycles (the symptoms being provided as data during the first cycle).

These hypotheses relating to functions may be associated with probable causes of malfunction, e.g. hypotheses (capacitor ([N5, N8]), C3, short-circuit).

2.3.2 Application of Rules of Selection

The system then activates in the back-chaining mode the rules of selection for the purpose of determining among all the hypotheses of previous cycles remaining to be examined that which the system will attempt to validate during this cycle. In the absence of remaining hypotheses, the rules select from the circuit functions not yet validated. This avoids an excessively rigid in-depth search strategy.

2.3.3 Validation

Validation of the selected hypothesis is based on the rules of validation activated in the back-chaining mode. These rules are conceptually similar to those of selection (since it is a question of selecting the method of validation) and involve the rules of operation and the rules of malfunction also used in the back-chaining mode.

Thus if the selected hypothesis defines a case of malfunction of a function by means of the rules of experience, the system must attempt to confirm definitively this malfunction by using the associated rules or, if this is not possible, must try to show that the function is correct. If the hypothesis expresses doubt regarding a function without defining the cause, the system acts in the opposite direction, attempting firstly to confirm correct operation of the function before considering its possible malfunction.

The validation result is conserved (function recognized as correct or incorrect or validation attempt failure, i.e. doubt concerning the behaviour of the function). It is used in subsequent cycles (rules of experience) as are any faults confirmed or detected by the rules of malfunction, e.g. fault (collector_base_junction_open_circuit, T6).

2.3.4 Processing of the Validation Result

The program reinitiates the troubleshooting cycle. The data acquired during the previous cycle are then used by the rules of experience.

The various stages of reasoning used by the rules (hypotheses concerning the functions and possible causes, validation or invalidation of a function, normal or abnormal operation of a function, type of fault) are recorded in a current fact base continuously updated by the rules (additions or removals) while information acquired as a result of measurements (voltages) and considered as definitive are store in the frames.

2.4 END OF DIAGNOSIS

This occurs :

- either at the start of a cycle when a rule of experience indicates that a defective atom or link has been found by a rule of malfunction during the previous cycle, the fault then being identified,
- or at the moment of selection when all the functions of the circuit have already been subjected to the validation phase, in which case the system has failed to identify the fault and can at the most propose the changing of non-validated atoms.

2.5 TRACE AND INTERFACE

A graphical interface (under IBM GDDM system) displays all or part (zoom) of the circuit to be tested.

Beneath the graphical field in which the circuit diagram appears, there are two alphanumerical fields for system requests and user replies.

When a test_point measurement is requested in the first of these fields, this test_point flashes in the diagram, enabling it to be easy recognized.

The execution of reasoning may be followed graphically by means of a colour code controlled by the rules, identifying at each cycle, the hypotheses generated by the rules of experience and then the attempt to validate operation or malfunction of the selected function.

Each time information is entered via the keyboard, the trace mode may be selected (or abandoned) at will for showing execution of the rules (see § 3.2). In any case, a complete trace of questions, answers and rule activations is available in files at the end of the session.

CHAPTER 3 - THE KNOWLEDGE REPRESENTATION

In DEDALE knowledge is represented in the form of frames and rules.

3.1 FRAMES

3.1.1 Description

Frames are used here for describing the factual knowledge of a circuit (another possibility would consist in formulating them in object-oriented language).

The conventional representation in frames is given in the form of quads
< object, attribute, facet, value >.

From the semantic point of view, it is possible to distinguish :

- the models which are special frames representing concepts, assembling here the knowledge on an electronic circuit in general (valid for all troubleshooting)
- the instances which are precise objects materializing a concept, here the knowledge relative to the particular circuit under test.

The attribute plays a special role : the ISA link connecting a model to a more general model or connecting an instance to a model. This hierarchical link automatically implies heritage of the values of the attributes of a model by its descendants. Thus a graph is obtained, the leaves of which are instances.

3.1.2 Attributes

Here are examples of attributes together with the highest-level object where they are defined.

Object	: any (represents any attribute) ; priority (a priori vulnerability coefficient)
Structure	: reference(s) with respect to a local reference frame ; form ; dimensions; structural_type ; test_points ; technology
Block	: composition (in blocks or atoms)
Internal_node	: composition (in internal_nodes or links)
Circuit	: defective_inputs ; defective_outputs
Test_point	: position(s) (with respect to a local reference frame) ; voltage (potential difference (pd) with respect to a ground reference test_point) ; pd(T) (potential difference with respect to test_point T) ; AC_voltage : AC_pd(T)
Node	: composition (in internal_nodes) ; test_point
Function	: sub_functions ; functional_type (e.g. the name of the function without its parameters)
Active_block	: temperature_compensation
Resistor, Inductor, etc.	: tolerance
Diode	: maximum_current ; maximum_reverse_voltage, etc.

It should be noted that the attributes, in the same way as the objects, may consist of a name or a name with associated parameters (Prolog predicate).

Many of these attributes, attached, as shown above, to their conceptual entities, are in fact used only at the level of instances and receive their values at the time of data acquisition on the circuit. Others (voltage, pd(T)) are used in the course of troubleshooting under their procedural attachment aspect (facet if_needed).

3.1.3 Facets

Contrary to domain-specific attributes, facets are more less universal in the frame representation.

- **Value Facets**

  Facets intended for receiving the value or values of an attribute (these values generally being objects, Prolog atoms, numbers or character strings) : VAL and DEFAULT (implicit information to be taken into account if there is nothing in VAL).

- **Filter Facets**

  Filter facets, which are constraints on allowed values of an attribute as well as filtering in the methods of searching for these values :

. DOMAIN specifies the domain of attribute value definition :

* This Domain may be described explicitly, e.g. as a list of entities : <capacity, technology, domain, {polarized, unpolarized}> or numerical interval : <capacity, tolerance, domain, between (1, 10)>.

* It may be described by using the object as variable ; it is then calculated when the object is instanciated : e.g. <circuit, defective_outputs, domain, outputs (*self*,   domain)>, in which outputs (self, *domain*) is a program producing the list of outputs (*domain*)   of the initial object self.

* It may be described implicitly as all the values confirming this property : V P(V)   . It is then not calculated in advance but any proposed value is accepted only if it confirms the property : e.g. <object, priority, domain, integer(V)> to indicate that the priority must be integer.

* Finally, the *domain* may have the names of frames as value (see exclusive_choice below).

. TYPE can assume the following three assignments :

* Single-valued when the value of the attribute is single, *which may be of the list type* as in
  <node, test_ point_list, TYPE, single-valued>
  <node, test_point_list, VAL, [T1, T5, T7]>.

  The single-valued character of an attribute implies restrictions when searching for a value (stop on the first value found) as well as for adding a value (error as soon as a different value exists). This is the type by far the most used in DEDALE.

* Multi-valued when the attribute can have several values, provided they belong to the domain
```
e.g. : <node, test_point, TYPE, multi_valued>
       <node, test_point_list, VAL, (T1, T2, T3)>
```

* Exclusive_choice when the attribute can have several values divided into classes with an exclusive character within each class.

. PRE_CONDITION acts as a filter in the search for a value of the attribute. Its value is an action (representing conditions to be satisfied) which will be executed before activation of an reflex if_*needed* found higher in the ISA hierarchy. This facet is for future extensions.

- Reflex Facets

Reflex facets constitute the active part of frames under which are stocked procedural knowledge, the initiation in cascade of which may be considered as basic reasoning within the frames.

. IF_NEEDED : this reflex, the value of which is any action, is initiated when searching for a value of an attribute (when there is nothing under facet VAL).
```
e.g. : <test_point, pd(T), IF_NEEDED,
                    pd_calculation (self,t,V)
         if not request (self,pd(t),V)>
```

Self indicates the initial object of interest (in this case a particular test_point), V is the value sought (here the pd of the test_point considered with respect to test_point t), pd_claculation if for example a Prolog program calculating the potential difference between two test_points, **'request'** is a frame utility function used for questioning the operator in order to obtain a value of an attribute or object.

. IF_ADDED : the value of this reflex is any action initiated when adding a value (under facet VAL) to an attribute : e.g. <test_point,pd(t), IF_ADDED, **put** <t,pd(self),VAL,-V>>, **put** being a frame utility function filling an object attribute facet with a given value.

. VALUE_ERROR : contains an action initiated when obtaining a value rejected by the filters (e.g. a value not belonging to the domain).

```
e.g. : <object, any, VALUE_ERROR, write ('the value', V, 'is erroneous. Give another value
         of', ATT, 'of', self)>.
```

. EXPLANATION : contains an action (in general a message) initiated when requesting an explanation from the operator following a question asked by the system.

- Utility Facets

. PROMPT : contains an action (in general a message) activated by the **'request'** function.
```
e.g. : <test_point, pd(t), PROMPT,
             if <ground, reference, VAL, t>
                then write ('measure the value
                    of the DC voltage on', self)
                and flash (self)
                if not write ('measure the value
                    of the DC pd between', self, 'and', t)
                and flash (self)
                and flash (t)        >
```

. PROMPT_EVAL : contains an action (in general a message) activated when evaluating a relationship (binary operator) between an unknown value of an attribute and a given value of the same type (such an evaluation often occurs as premise of an inference rule and constitutes the interface of the rules/frames type) Ex. : T1 ; pd(T2) ; < ; 3.

. COST : allows information acquisition cost data to be placed here (value of an attribute) by means of an IF_NEEDED procedure (e.g. accessibility of a test_point or a measurement). This facet is for future extensions.

3.2 RULES

3.2.1 Description

These are rules of the type : **if** premises **then** conclusions, reflecting the inferential knowledge on troubleshooting at the level of the rules of electronic operation or malfunction, rules of experience or "meta-rules" of strategy.

These rules are written in external form as Prolog terms, with a certain number of keywords (rule, if, then, and, or, no, among, etc.) defined as prefix or infix Prolog operators with conceivable priorities for reflecting the syntax adopted. The variables are Prolog variables. This defines an external syntax, easily modifiable during the design phase while remaining within the Prolog framework. Thereby it allows direct acquisition of the rules by the Prolog interpreter for the purpose of translation into internal rules usable by the inference engine.

Unification is used to the fullest extent, the conditions (premises) and actions (conclusions) being evaluated either in the system of frames or in the base of current facts or as general Prolog goals.

3.2.2 Forward chaining

Forward chaining is performed by saturation on the current fact base, i.e. by enchaining passes through all rules until the fact base is no longer modified by such a pass (optimization ensuring that, during a pass, only those rules for which one premise at least has been modified during the previous pass, are activated). This check can be easily extended to saturation on the knowledge base constitued from the frames. In an open world, an additional check is performed that no contradiction occurs in the fact base.

Other control modes are possible, such as the interruption of forward chaining on a particular activated rule (presence of a STOP as conclusion of this rule).

A check may be executed by certain rules of the packet (operating as meta-rules). It is in this manner that the action of an activated rule may be to inhibit temporarily (during remainder of the forward chaining) or even to definitively remove a certain number of rules from the packet.

During forward chaining, it is possible to inhibit at will questions to the operator or more generally the procedural attachments IF_NEEDED.

Example of a rule of experience used with forward chaining :

```
rule reference_voltage_fault
  if fault (power_supply,*N)
    and *N;block_connected;*B
    and exec (presence (*B,*Z) & function (*Z,zener,*Func))
    and non-validation (*Z)
  then hypothesis (*Func,*Z,forward_biased).
```

It indicates : if there is a power supply problem on a node and if a block connected to this node comprises a Zener diode not yet validated, then adopt the hypothesis that this Zener diode is forward biased (fault, validation, hypothesis are predicates of the current fact base, block_connected is a frame attribute, presence and function are Prolog predicates).

3.2.3 Backward chaining

3.2.3.1 Mechanism

In the case of backward chaining, a rule is initiated when a goal (initial Prolog goal initiated from the calling "program" or sub-goal from a premise of another rule) "unifies" with an initiator of this rule. In general, any conclusion of a rule, other than the call to execute a specific program (exec (**<**Prolog-goal**>**)), i.e. any conclusion of the **<fact>** type (and **non <fact>** in an open world) or **<action_frame> = <object>;<attribute>;<value>** can operate as initiator of this rule : the "external" rule is thus converted during compilation into as many "internal" rules (Prolog clauses) as it possesses such conclusions. The rule, initiated in this manner, evaluates its premises (possibly processed as other sub-goals) and then, in the event of success, activates its conclusions (validating in particular that having served as initiator). Implicitly this backward chaining is performed by the Prolog interpreter, i.e. the strategy of choice among several rules having initiators being unified with the same sub-goal, is to initiate these rules in the order in which they are stored in the working space.

When this sub-goal possesses no free variables, no back-track point is created at its level. On the other hand, when this sub-goal possesses free variables, a back-track point is created, i.e. the possibility of several validations of this sub-goal corresponding to different variable instantiations is left open.

Example of a rule of operation used with backward chaining :

```
rule diode_func
  if *Anode;pd(*Cathode);*U
    and *Cathode;current;*I
    and (1 among exec (between(0.5,*U,0.8)&*I=0)
              or exec (le(*U,0.5)&*I≈0))
  then normal_operation (diode([*Anode,*Cathode]),*D).
```

It indicates : if the potential difference between the anode and cathode of a diode lies between 0.5 V and 0.8 V and if the cathode current is clearly non-zero or if this potential difference is less than 0.5 V and the cathode current is close to zero, then diode operation is normal.

A sub-goal initiating this rule is for example : normal-operation (diode([N6,N2]),D3) (normal_operation being a predicate of the fact base ; pd and current being frame attributes, between, le, =, ≈ being Prolog predicates or operators).

3.2.3.2 Strategies

- A strategy at the level of the order to evaluate the premises of an initiated rule may be envisaged.

The Prolog by-default strategy (evaluation in the written order) is often not the most pertinent in a given context. For example, there is no purpose in a conjunction of premises, in initiating in-depth search (with possible questions to the user when the answers have no practical use elsewhere) for the first (n-1) if the nth is clearly impossible.

A first rough examination of the premises can thus be made, thereby resulting in the validation or invalidation of some and therefore sometimes of the whole rule. Only in the case of indecision is a more complete examination required with in particular initiation of the procedural attachments IF_NEEDED in the frames stored during the first pass. During this reexamination, it is also possible to involve an order taking into account the cost of acquiring the information contained in the IF_NEEDED attachments (use of the COST facet).

- Another strategy can be used at the level of the order in which the rules likely to lead to a given sub-goal are examined (i.e. rules whose initiators unify with this sub-goal).

Here again, it is often preferable to avoid the order given by Prolog, which is that in which the rules are loaded ; this is done for the purpose of separating the declaration of knowledge (declaration of the rules by the expert) and the control of their use (which the data-processing engineer should be able to *specify independently) as well as for the purpose of achieving maximum declarability and modularity (rules* produced "pell-mell", possibly by several persons).

3.3.3.3 Trace

In the case of backward chaining, a trace (dynamic on the console and recorded in files) is provided of :

- the initiation of each rule on a sub-goal,

- the success or failure of this rule : in the latter case, the premise on which this failure occurred is first mentioned.

This trace is indented according to the depth level of the calling sub-goal. In addition, the list of rules effectively executed in backward chaining for an initiated goal is provided. This is used, for example, in the basic DEDALE cycle in order to select (for forward chaining) the rules of experience having at least one premise unifying with a conclusion produced by a rule executed during the previous back chaining.

3.2.4 Representation of Rules

In addition to their internal translation into Prolog clauses, the rules have a frame representation. This makes it easy to talk of rules by means of some of their properties (attributes), which is indispensable, for example, when writing the strategy and control meta-rules. Among others, a rule will therefore possess premise and conclusions attributes producing the list of its premises and conclusions. Thus it is possible, when compiling a packet of rules P1 (with backward or forward chaining), to determine the possible effects of a given rule on another packet P2 of rules (with forward chaining), thereby allowing chaining optimization : forward chaining of P2 limited to the rules of P2 using the new knowledge generated by the execution of P1.

CHAPTER 4 - MAN/MACHINE INTERFACE

In addition to the interfaces needed between the expert and the knowledge base (FRAMES and RULES) for constituting and updating this base, two other types of communication are provided, related to :

- general information on the circuit under test (expert domain),

- specific information on the circuit under test (operator domain).

A set of syntaxes defined by DEDALE allows the communication of information obtained from external files or from the console. A set of messages and graphical tools enables the user to communicate with DEDALE.

4.1 GENERAL INFORMATION ON THE CIRCUIT

DEDALE merely records the data provided by an expert an checks their syntax. The data are obtained from two files. The first file contains the structural description of the circuit and the second file contains the functional description. Semantic coherence at the level of block, node and function names is required in order to link or not structural and functional entities.

4.2 SPECIFIC INFORMATION ON THE CIRCUIT UNDER TEST

During diagnosis, DEDALE may require additional information for investigating or concluding : qualitative measurements or observations of electrical parameters, temperatures, physical appearance of components.

Example of a message : what is the potential difference between the anode and cathode of diode D1 ?

The expected reply may be checked (and possibly refused) by DEDALE. In this case, the message is repeated (generally with a message of explanation).

CONCLUSION

The present state of progress enables DEDALE to be considered as a prototype, but one already possessing performance characteristics such as it can be operated by final users : troubleshooters. This is mainly due to the quality of the IBM interpreter used (VM PROLOG) [6] and the power of the computer (IBM 3090/200) on which the system is run.

It is nevertheless necessary to industrialize the system which should result in easier maintenance of DEDALE and extension of the proposed functions.

This will be achieved by using the EMICAT (MI4) expert system development environment which ESD has produced for its own needs and which is now commercially available. EMICAT (MI4) is based on an extension of the Prolog language to an object-oriented language. The objects provide standard representation of all types of knowledge : factual, deductive (rules) and meta-knowledge. Among the many facilities offered, mention has to be made of the possibility of defining multiple-rule formalisms, of limiting both the applicable rules and the objects they may apply to, and of memorizing knowledge base 'states' used later for back tracking.

The contribution of EMICAT (MI4) is at the level of :

- maintenance, since the major part of maintenance will henceforth be at the level of the tool itself as a constituent element of the overall system and since the use of multiple-rule formalisms will considerably improve readability,

- functions, since the mechanisms offered (including in particular the state memorization mechanism) will allow the implementation of more sophisticated and more efficient strategies.

Another current evolution in the DEDALE system deals with qualitative modelling and reasoning. Since no numerical model of behaviour is available for components outside of their normal range of functioning, the chosen solution is to express models in terms of main qualitative changes in the parameters involved [4].

In conclusion, it must be emphasized that the cost of production-engineering an expert system remains high but acceptable as a result of using high-performance tools such as EMICAT (MI4) and is justified for applications amenable to the techniques of expert systems, such as diagnostic and maintenance. Several expert systems of this type are being produced or developed within ESD for areas as varied as battle-tanks, airborne radars and satellites.

REFERENCES

[1] Davis,R.,Shrobe,H.,Hamscher,W.,Wieckert,K.,Shirley,M. and Polit,S.,Diagnosis based on description of structure and function, Proceedings of the National Conference on Artificial Intelligence, Pittsburg, PA (August, 1982) 137-142

[2] De Kleer,J.,and Williams,B.C., Reasoning about multiple faults, Proceedings of the National Conference on Artificial Intelligence, Philadelphia, PA(August, 1986) 132-139

[3] Brown,J.S., Burton,R. R. and de Kleer,J., Pedagogical, natural language and knowledge engineering techniques in SOPHIE I, II and III, D. Sleeman and J.S. Brown (Eds.), Intelligent Tutoring Systems, (Academic Press, New York, 1982) 227-282

[4] Raiman, O., Order of magnitude reasoning, Proceedings of the National Conference on Artificial Intelligence, Philadelphia, PA (August, 1986) 100-104

[5] Brown, A.L., Qualitative knowledge, causal reasoning, and the localization of failures, Artificial Intelligence Laboratory, TR-362, Cambridge, M.I.T. (1976)

[6] VM/Programming in Logic, Program description and operations manual, IBM (1985)

AN EXPERT SYSTEM FOR AIRCRAFT CONFLICT RESOLUTION IN DENSE AIRSPACES

by

B. A. Bowen, Ph.D., P. Eng.,
Professor, Department of Systems and Computer Engineering
Carleton University, Col. By Drive
Ottawa Canada, K1S 5B6
and
President, CompEngServ Ltd., 265 Carling Ave., Suite 600
Ottawa Ontario, K1S 2E1

SUMMARY

A hybrid knowledge-based system is described which provides advice to Air Traffic Controllers on the optimal tactics for resolving predicted aircraft conflicts. The overall functional architecture is described which has both computational algorithms (in classical software) and rule bases containing the knowledge and experience of controllers in the air traffic environment.

The system is designed to replicate the way in which a Controller might react to a predicted conflict(s). It responds to conflict predictions with resolution advice, which depends on both formal rules (e.g., MANOPS) and on heuristics obtained from Controllers based on their experience in the air space.

The hybrid nature of the system is based on a design approach involving a decomposition of the overall sequence of logical decisions required to resolve the predicted conflict into a "Global Inferencing Algorithm" (GIA). The GIA approach simplified the design, partitioned in a natural way the knowledge base, enhanced the performance, and permitted a systematic validation of each step of the resolution by comparison to a Controller's decision in similar circumstances.

This paper describes first a GIA for resolving conflicts in a sparse airspaces (i.e., one in which a resolution is effected without inducing further conflicts) and then extends this to a dense air space (i.e., one in which a resolution tactic could induce subsequent conflicts). The question of convergence (and/or the lack of it) of a completely autonomous system is discussed but not resolved.

The prototype of the sparse airspace system was programmed using an inductive systems builder called TIMM on an IBM/XT. It contains 750 rules and responds to presence of a detected conflict in approximately ten seconds.

0.0 A Perspective on Air Traffic Control

Air traffic controllers provide the essential strategic and tactical control point in a complex system designed to ensure the safe orderly movement of aircraft. The job requires considerable intelligence to perceive the essential aspects of impending dangerous situations, and considerable experience to make satisfactory decisions to avoid conflicts in flight paths, under situations which at times can be very stressful. The spectre of disaster always haunts the controllers as the potential for errors is contemplated. Air traffic control requires decisions involving human lives at one extreme, and considerable capital costs and/or operating costs at the other; all this with incomplete data, in an imperfect control loop, and in a time constrained situation.

Aircraft flying in controlled spaces are assumed to have filed flight plans, and normally to be under the surveillance of ground based radar or at least periodically to file position reports. The pilot's and the controller's actions are constrained by many factors which could be considered as being based on both static and dynamic data. Static data is slowly varying and includes such items as government prescribed rules for separation, the physical capabilities of aircraft, the location of emergency fields, features of the terrain, etc. Dynamic data includes such things as the current weather, emergency landing priorities, the number of aircraft in the controlled space, etc.

Errors in judgement calls can occur from many uncontrollable factors in the process, such as imperfect understanding of, or adherence to instructions by the pilot, lack of exact information on location, poor and/or noisy communications channels, unexpected weather, etc. Thus, the controller must work with an intrinsic uncertainty both in input data and with exact compliance to the control decisions.

A conflict occurs whenever a given airspace is occupied by two or more aircraft, and is defined as the violation of any one of a set of separation criteria specified in the appropriate Air Traffic Manual of Operations (MANOPS). Conflict resolution consists of developing and implementing a set of maneuvers specifying flightplan modifications for one (preferably) or more of the aircraft involved in the conflict. This set of maneuvers is derived by applying some combination of the possible resolution tactics to the particular conflict being considered.

In general, conflict avoidance is an n-body problem, involving the separation of moving objects in a three-dimensional space. This class of problems exhibits an exponential growth in computational complexity as the number of bodies increases. Thus, classical programs based on algorithms have failed for all but very simple problems. Since this problem is routinely solved in practice, a heuristic approach based on the experience of Controllers seemed appropriate.

1.0 Introduction

Air traffic control has many operational procedures and other attributes which must be learned from experience. This single feature has influenced many research groups world-wide to attempt to incorporate some aspects of artificial intelligence into the automation process.

In this paper, we will review the conflict resolution problem and demonstrate some results with respect to automating this task by capturing the rules and procedures used by controllers to arrive at a decision. As part of this demonstration we will show how the process of executing such a task can be decomposed and allocated to regular software as well as to rule-based procedures. The result is a hybrid software system containing both conventional software procedures and modules containing the rules representing the procedures followed by an experienced controller.

To accomplish this demonstration, we first discuss the resolution in an airspace that we refer to as sparse. In this simplified air space, the conflict resolution can take place without inducing other conflicts. This assumption allows the design process to go on without undue difficulty and provides the tutorial background for understanding the more complex space in which the resolution of one conflict can induce others.

The resolution of conflicts in a space containing N aircraft can become computationally complex very quickly. Indeed, the reason that algorithmic attempts to solve this problem have failed is a function of both the exponential complexity of the computational load that these routines demand and the difficulty of defining and implementing the inferencing process required. This is, perhaps, the main motivation for using an expert system approach; for controllers routinely solve this problem based on experience without having the benefit of a mathematical algorithm. Controllers normally adopt the procedure that is simplest to execute. A good resolution is the simplest to implement tactically. This implies also that only one aircraft should be involved whenever possible.

While an expert system will always attempt to emulate the final decisions of a Controller, it is often possible to consider alternatives that would be computationally intense without the aid of computers. Thus an expert system backed by reasonable computing power can consider alternatives and procedures that might be eliminated in normal operations. There is no evidence that the addition of this form of automation has increased the quality of decisions, and this is not the major issue. The issue is, finally, can an automated system handle traffic on arbitrary flight paths in dense spaces with speed and reliability.

Many attempts have been made to model the air traffic problem by analytic means. It appears that these attempts have been in three areas: either algorithms, inferencing models or expert knowledge. The most promising in our opinion is based on experience, combined in an overall inferencing procedure that is dependent both on rules and on procedures. Thus, our approach is to start with 'What' must be done and let the 'How' emerge either as a knowledge base or as an algorithm. This approach seems to drive the requirements for particular functions and emulates what a Controller does in practice. The question of implementing the "Hows" will be discussed and illustrated.

In Section 2.0, a simplified model of the ATC problem is proposed so that we can agree on the basic jargon for further discussion. The description is incomplete, but sufficient for our work.

In Section 3.0, a global set of interconnected modules is developed which solves the resolution problem in a sparse airspace. This is called a global inferencing algorithm (GIA). Some of the modules are classical software and some contain rules obtained from controllers. A system was programmed from this model and results are presented.

In Section 4.0, the system is extended to cover the case where a resolution procedure can induce further conflicts which in turn have to be resolved.

2.0 Modeling the ATC Problem

The requirements definition formed the initial parts of the systems design. Such a definition consists of feasibility, attributes, performance, maintenance and growth, and validation sections.

The feasibility of an expert system implementation for conflict resolution was based on the evaluation of a number of specific attributes. For example, the conflict avoidance task is well bounded in terms of the knowledge required and results obtained, and makes extensive use of symbolic reasoning in terms of the spatial and temporal relationships between moving aircraft. The task itself is routinely taught to new controllers, and is, therefore, decomposable into sub-tasks for this purpose. The range of problems that can be solved and the quality of decisions increases with experience. Used as an advisor, the system would alleviate much of the pressure felt while handling busy airspaces. An expert system would also allow controllers to apply a high level of capability to conflict situations and to prepare alternate solutions for consideration. Expert solutions presented by the system would also provide on-the-job skill improvement for novice controllers.

These factors provide sufficient evidence to warrant an expert systems approach to the conflict avoidance problem, an assertion which will be validated by the design and performance of the prototype system. The prototype was, therefore, designed to demonstrate both feasibility and proof of concept.

The prototype's attributes can be distinguished as being either physical or logical. Physically, the prototype runs on an IBM PC/XT. Logically, it used a commercial system builder (called TIMM) to create and maintain the knowledge bases. The user interface, which prompts the user for data and presents formatted results, was written in a high level language (Fortran), so that a user need not be familiar with the expert system builder to run the prototype.

To demonstrate feasibility the prototype implemented a subset of all conflict types, tactics, and domain data elements. These provided a sufficient foundation for proving feasibility, and for demonstrating the concepts relevant to conflict avoidance. The architecture handles single conflicts, and does not consider future conflicts that might be caused by resolving present ones. Growth of the prototype is possible, because it is capable of being expanded to include more conflict types, resolution tactics, data on which to base decisions, and rules through which decisions will be made. In addition, as will be shown, it is the first step in the extension to dense spaces.

Validation was performed by devising test problems that exercised all of the system's capabilities and each of the conflicts and tactics recognized. The solutions given by the expert system were compared with those given by a domain expert in conflict resolution.

This system emphasized expert system technology for the selection and development of conflict avoidance tactics. For this reason, it assumed that conflict detection is handled by standard procedures callable at any time. It also did not attempt to solve classical software problems, although they were identified and characterized. This included automatic acquisition of data from present air traffic systems, and calculating details of the maneuvers required by each tactic.

3.0 A Global Inferencing Algorithm for Sparse Airspaces

In this section, a global inferencing algorithm is presented for conflict resolution in a sparse airspace. The purpose is to both illustrate how this procedure is done and to prepare for the extension to a dense airspace. The procedure is first to interpret and then to replicate the detailed steps taken by the airtraffic controller in resolving a conflict. The knowledge engineer must follow the controller's thought processes and attempt to represent them in a form that is codeable by current technology. The process requires extensive cooperation, as both the controller and the knowledge engineer must make accommodations to each others skills and technology.

System Level Issues

Conflict resolution for our purposes is viewed as a potential reconfiguration of the aircrafts' original flightplans. This view is taken because it is possible that the resolution of a conflict might permanently modify the remainder of the flightplans of the aircraft involved. Also, more efficient and more general resolutions can be achieved if flight plan reconfigurations are allowed.

Before a global inferencing algorithm can be created, the logical sequence of functions must be postulated. The first step in finding these functions is to view the Conflict Avoidance Expert System at it's highest level of abstraction as a black box, as shown in Figure 1. It accepts some specification of an airspace as it's input, and provides a recommendation for avoiding the conflict as it's output. It also requires rules, heuristics, and classical software procedures to guide the system's operation, which must be acquired from air traffic controllers and aviation authorities during system design.

The input data includes at least the same information available to an air traffic controller: for example, some statement of aircraft flightplans (analogous to the flight strips currently in use), and data about the airspace, which could be extracted from present-day air traffic computer systems. The output data specifying conflict resolutions must be made available far enough in advance of the conflict in order to be used, so the conflict detection must be done well before the conflict, and the Conflict Resolution system's response time must be reasonably short. The overall time budget allocated to each function is a system level consideration. All of this data can be formalized and tabulated in a data dictionary and as a set of performance specifications.

Global Inferencing Algorithm

Initially at least, three logical functions seem necessary for resolving conflicts, as shown in Figure 2. First, the conflict must be detected and characterized. Second, an acceptable set of tactics that may be used to solve the conflict must be selected. Finally, the details of each tactic's implementation must be determined for the particular conflict and airspace state.

The three functions shown in Figure 2 must answer the following questions:

Conflict Classification

What kind of conflict is it?

Determine Tactics

What tactics might be used to resolve this conflict?
Which of these will actually work?
Which tactic is the best?

Determine Implementation

How will these tactics be implemented?

Answers to the first question classifies the conflict. It ensures that the system will understand the type of conflict it is dealing with, and that resolution tactics to be applied to this conflict may be enumerated. Data required to answer this question will include only information about the two aircraft involved.

Answers to the second question select tactics on the basis of data about the conflict itself and the aircraft involved. This is analogous to the way an air traffic controller first focuses attention on the conflict, only widening the scope of attention when it is determined what might be done. The result of this step is a set of potential tactics, (i.e., those tactics that have the potential to resolve the conflict, depending on the state of the airspace).

Answers to the third question broadens the scope of the system's attention to include the airspace and other aircraft in it, so that each potential tactic must result in a safe conflict resolution or be discarded. This evaluation results in a reduced set of possible tactics (i.e., tactics with which it is possible to resolve the conflict).

Each of the possible tactics is now examined to determine which one is "best". This prioritization must take into account each aircraft's efficiency of operation and any operational constraints imposed by air traffic procedures.

The final question is "How would these tactics be implemented?", and is answered by determining the exact maneuver specifications for each possible tactic.

Each of these functions is treated in more detail in the following:

Conflict Classification

While detailed definitions for all possible conflict situations are contained in documents such as MANOPS, Controllers typically organize related conflict situations into groups, and solve each group, or "type" of conflict, in a similar manner. However, conflict detection software is based on strict conflict definitions. Therefore, the system must classify detected conflicts (i.e., determine their type, as a controller would, in a way that the rest of the system will be able to use).

Each conflict type will have a set of resolution tactics associated with it. Therefore, the organization of the conflict types and their resolution tactics is interdependent, since each detailed conflict must be solvable using any of the associated tactics, regardless of how other factors influence the conflict's resolution.

The conflict classification function in the prototype consists of a single rule base containing [illegible] rules that will recognize 10 types of conflicts. It requires basic data about the conflict, including the flight level, airspeed, heading, and attitude of both aircraft at the moment of the conflict, all of which may be derived from data available to controllers, such as flightstrip information. The prototype presently carries three of these conflict types through to Prioritization: Head-On, Overtaking, and Crossing Tracks.

Tactic Selection

The purpose of this function is to determine those tactics that have the potential to resolve the specified conflict. This selection process is based only on knowledge about the conflict itself (i.e., on data about the conflict type and the aircraft involved). Data about the airspace itself or other aircraft are not used.

The knowledge base can be partitioned for efficient searches by using a look-up table to determine which resolution tactics should be explored for a given conflict type. A conflict type accesses a row in the table, which specifies which tactics should be explored. A subset of this look-up table is shown in Figure 4.

The resolution tactics that are applied by the prototype are Change Flight Level Up, Change Flight Level Down, Increase Groundspeed, and Change Track to Port. These tactics are applied to both aircraft as appropriate. The tactic selection expertise consists of a rule base of 111 rules, and has been partitioned according to both conflict types and resolution tactics. This will result in the Tactic Selection rule base being partitioned into a number of rule sets, with one set for every combination of conflict type and resolution tactic, as shown in Figure 5.

Tactic Evaluation

When given a set of tactics that could potentially solve the conflict, the Tactic Evaluation function must eliminate those tactics that can not be used to solve the conflict in the present airspace configuration. The result is a set of possible tactics (i.e., those tactics for which it is possible to determine a complete conflict resolution procedure).

This evaluation involves considering all other factors that might affect the implementation of this tactic. This would include the location of other aircraft, possible interferences caused by planned maneuvers, airspace restrictions, maximum cruising altitudes, etc. In order to determine some of this data, classical software routines must be invoked by antecedents within the rule base.

The Tactic Evaluation rule base, consisting of 72 rules, is partitioned in the same way as the Tactic Selection rule base, so that only rules pertaining to the individual problem will be examined.

Tactic Prioritization

This function will accept a set of possible tactics and sort them in order of preference by assigning a priority to each tactic. These priorities specify their relative preference, rather than their absolute preference. It uses a rule base of 66 rules, partitioned in the same way that the two previous functions are.

The calculation of these priorities is based on each tactic's base priority (which states it's initial utility unconstrained by any conflict factors), operational preferences of the airspace (e.g., the effects of NOTAMs), and efficiency criteria of aircraft involved (e.g , fuel consumption).

Resolution Procedure Development

The resolution procedure development function is responsible for calculating the details of all maneuvers required to resolve the conflict. In order to do this, preceding modules must provide this function with all of the information required to calculate the maneuver specifications. This required data is different for each tactic, and depends on the number of maneuvers in the tactic, the characteristics of the conflict situation, etc.

It is assumed that the conflict and the airspace will impose time constraints on the maneuvers, forcing them to occur within certain time windows. In order to produce unambiguous specifications, it is necessary to use both distance and time factors, depending on how the flight of aircraft is affected by wind conditions, navigational capabilities, etc.

This function is a problem in classical software, and is essentially a set of maneuver vectors specified in terms of time and space. It can be completely specified as a set of procedures, requiring no heuristic knowledge, and is considered to be in the same class as, although less complex than, conflict detection. Therefore, it was not implemented in the prototype system.

Validation Results

Validation testing was performed by providing problems for the system to solve that showed correct operation and demonstrated particular aspects of it's capabilities. These results were approved by domain experts as corresponding to those they would have recommended.

An example of a test problem solved by the system is shown in Figure 6. It is an overtaking-type conflict between a C5A Galaxy, representative of a large and cumbersome transport aircraft, and a Cessna, a small and more maneuverable aircraft. At its current level of expertise, the prototype knows about applying five tactics to this conflict, which, along with the prototype's results, are shown in Figure 7.

Performance Assessment

The prototype presently requires approximately 10 seconds to resolve a conflict, not including time required for user interactions. Given the automated acquisition of data (to free the controller from this time-consuming task), the prototype could be considerably upgraded before it's performance required a machine more powerful than an IBM PC AT. This growth will eventually reduce performance below acceptable levels due to the large number of combinations of conflicts and resolution tactics. However, proprietary developments at CompEngServ suggest that a full scale conflict avoidance system can be supported by an IBM PC/AT with an attached real-time expert system processor.

The classical software portion of the prototype is the shell, which contains 1,100 lines of Fortran code. However, the growth of the shell's complexity would be less than linear, as the shell presently handles knowledge base invocation, which would not change, and data input and output processing, which would expand slowly since most of the system's growth would be confined to the knowledge bases.

4.0 A GIA for Dense Airspaces

The extension of the inferencing procedure to a dense airspace is demonstrated here in two steps: First, to a situation in which the Controller is still involved in resolving difficult cases (called semi autonomous) and finally to a situation in which the system searches for a final solution and calls for help only when no solution seems possible.

The design of a such system is dependent on the philosophy adopted. In the approach adopted by CompEngserv, it was considered important that any resolution procedure disturb the airspace as little as possible. This implies that any tactics adopted to resolve a conflict should, if possible, disturb only one of the two aircraft involved and only in extreme circumstances should the other (or others) be disturbed. Clearly the application of this rule must be dependent on the experience of controllers, since the boundaries are not well defined and any procedure must always include safety as the prime directive.

The other important consideration is how much autonomy is the system to be granted. In the end, this would depend on the confidence level, which would grow as the system was seen to solve more complex problems. This issue is not addressed here, but the technical problem of the convergence of the resolution algorithm is postulated. The system being proposed has not been constructed (at the time of writing), and thus the convergence can only be argued.

The overall design proceeds in a sequence of steps (to resolve the conflict), with each step being based on the principle of minimal perturbation of the existing state space. We will present the material as an extension to the sparse space, and in subsequent evolution from semi autonomous to autonomous. The semi autonomous system is defined as one in which a final recourse to help from the controller is always an option. In such a system the question is when to call for help. In an autonomous system no such help is available. In such a system, the question of convergence is upper most. In fact, we show a final call for help when the resolution procedures begin to induce an increasing number of conflicts. Whether this will happen or not is indeterminate.

Consider first a semi autonomous system:

The global inferencing algorithm for a semi autonomous system is shown in Figure 8. The algorithm proceeds in three steps. The approach is to make every effort to find a resolution procedure which will not disturb other aircraft (i.e., not induce other conflicts). In this situation, the short list of candidates, produced under the assumption that the space is sparse, is assumed to contain a possible induced conflict. The list is submitted to a detection procedure which extends the solutions to predict further conflicts. If at least one solution exists for which no further conflict is induced, this is presented to the Controller.

If all possible tactics based on the original constraints induce further conflicts, the system relaxes the constraints and searches for new solutions. If some exist, these are presented to the Controller. If not the Controller is asked for help.

Consider now the extension to the autonomous controller:

If the Controller is not included in the Help loop, then the system must define the response by further processing. An inferencing algorithm for this is shown in Figure 9. The first procedure orders the original short list to determine the least critical induced conflict. The partitioning algorithm is assumed here to create the minimal perturbation on the aircraft flightplans in the airspace. Under this assumption, the present conflict is resolved ignoring the induced conflict(s); following which, the induced conflict is then resolved by following the original procedure. It is assumed that the definition of 'critical' is such that induced conflicts can be resolved within the response time of the system.

The procedure executed in the first step of Figure 9 could eventually terminate with a null set indicating that, within the defined partition of critical and noncritical conflicts, no solution can be found. It is assumed at this point that the situation has become critical and strategic considerations must be abandoned for short term tactical procedures to insure safety.

The procedure at this juncture is to abandon the minimum perturbation criteria and partition on the bases of safe passage. The algorithm in this case is recursive and is best shown as pseudo code. The approach shown in the pseudo code in Table 2 is as follows:

The list of preferred tactics is examined one by one.

For each one, the airspace state is computed based on the projected execution of the tactic.
The number of induced conflicts is then computed. The critical assumption made in what follows is that if the number of induced conflicts increases the solution is abandoned.

The procedure continues until all alternatives are exhausted and then calls for help.

The convergence of this algorithm has not been tested against real data.

5.0 Summary and Conclusions

The procedure as presented was shown as a linear sequence of activities. In fact, there is considerable parallelism in the algorithm. The parallel form, using structure diagrams (from the Ada language [11]), can be used to find the parallelism. Parts of the algorithm can execute, even though the results may not be used. Thus, for example, in Figure 8, the second phase can be started as soon as the selection process for the short list is completed. It is obvious also that the third phase can be started as soon as the conflicts have been identified. A fully interconnected structure diagram to exhibit all the parallelism is straight forward to develop. The results offers encouragement that the computations exhibit sufficient parallelism so that resolution tactics could be obtained in a relatively short period of time (a few minutes) for reasonable cost (few PCs)

The assumption of resolving conflicts with a minimal perturbation of the airspace is arguable. For example, other starting assumptions could lead to different decision criteria and the order of processing. In particular, the assumption might change depending on the current density of aircraft in the space, or be permanently altered in a tight tactical air space such as around an aerodrome. For our purposes, it lead to a nicely definable resolution procedure.

The convergence of the algorithm is a matter of concern whenever autonomous conflict resolution is discussed. The algorithm as presented has not been tested against live airspace situations. Thus, the convergence is a matter of conjecture and not liable to mathematical proof. However, the resolution algorithm seems to follow the procedures used by Controllers, and intuitively should converge. In addition, sufficient parallelism exist so that reasonable computing power should yield advice in real-time. The question always remains however, as to a pathological situation existing in which the resolution procedure starts to converge. In the end, as shown, it seems probable that an appeal to a Controller must be built into the system.

The question of the accumulation of sufficient knowledge to fill the required knowledge bases is relevant. Again this question cannot be answered definitively, however, experience suggests that Controllers can supply such information and that rational decision can be made as suggested.

The implementation of this algorithm and hence the answers to many of the timing and convergence questions depends on the equivalent parallel representation of the algorithm. The implementation could then be carried out using a multiple processor system such as Intel's multibus or Motorola's VME bus, with an appropriate choice of processors, following the procedures outlined in [12].

Further work that is in progress includes the obvious requirement to test for convergence of the algorithm against real and simulate conflict situations. In the future, the parallel version of the algorithm can be constructed and a suitable architecture devised for implementation to create maximum response in a dense space. Finally, the partition of the data base and the optimum relations for each need further study.

6.0 Acknowledgements

The original work on conflict resolution in sparse airspaces was done while consulting for CompEngServ Ltd of Ottawa. This work was funded by the Transportation Development Centre of Transport Canada. It was first reported publicly in a paper coauthored by Mr. J. D. Beaton of CompEngServ (now with Esper Architectures of Ottawa) and Mr. N. M. Standen, Chief, Aeronautical Technology Development, Transport Canada, Ottawa. Mr. Pierre Sauve of Air Traffic Procedures Development, Transport Canada provided the controller expertise needed to fill in the knowledge bases and provided the test cases for validation of the sparse space resolution system prototype. Mr. D. G. Bowen of CompEngServ Ltd., provided extensive discussions of the extension of the GIA to dense airspaces based on his knowledge of the overall ATC problem.

Mr. Hubert Gervais, Operational Manager of the Flight Data Modernization Program (FDMPs) and his team offered valuable suggestions on the future systems requirements and the pragmatism of air traffic control. This assistance is gratefully acknowledged.

7.0 References

1. Bowen B. A., J. D. Beaton, N. M. Standen, "A Conflict Resolution Algorithm for Sparse Air Spaces using a Hybrid Knowledge-Based System," Canadian Aeronautical and Space Institute, Sixth Canadian Symposium on Navigation, Vancouver, B.C., May 1986.

2. Bowen B. A., J. D. Beaton, N. M. Standen, "Feasibility Demonstrated for Automatic Conflict Resolution," ICAO Bulletin, Sep. 1986, pp. 25-27.

8. Findlas N. V., "Air Traffic Control: A Challenge for Artificial Intelligence", AI Expert, Jan. 1987, pp. 59-66. (This paper contains an excellent set of references on inferencing procedures applicable to the ATC problem).

3. Zellweger, A., "Man-machine Interface Implementation in the Air Traffic Automation System," Proc. Eighteenth Hawaii International Conference on Systems Science, II (1985), pp. 338-348.

4. Zellweger, A., "Challenges in the Implementation of the Next Generation Air Traffic Control Automation System," Proc. COMPSAC 84 (1984).

5. Wesson, R. B., "Planning in the World of the Air Traffic," Proc. IJCAI-77 Cambridge. Mass., 1977, pp. 473-479.

6. Lo, R., and N. V. Findler, "An Examination of Distributed Planning in the World of Air Traffic Control," Journal of Distributed and Parallel Processing, Sept. 1986.

7. Findler, N. V., T. W. Bickmore, and R. F. Cromp, "A General Purpose Man-Machine Environment with Special Reference to A r Traffic Control," International Journal for Man-Machine Studies 23 (1985), pp. 587-603.

8. Shively C. A., "AIRPAC: Advisor for Intelligent Resolution of Predicted Aircraft Conflicts," 24th Annual Technical Symposium. Washington D.C. Chapter of ACM, Gaithersburg, MD., June 20, 1985, pp. 37-43.

9. Ball R. G., G. Ord, "Interactive Conflict Resolution in Air traffic control," Designing for Human-Computer Communication, Academic Press, London, 1983.

10. Cross S. E., "Computer Understanding of Air traffic control Displays," IEEE Transactions on Systems, Man and Cybernetics, Vol. SMC 15, No. 1, January/February 1985.

11. Buhr R. J. A., "Systems Design with ADA," Prentice Hall Inc., Englewood Cliffs N. J., 1984.

12. Bowen B. A., R. J. A. Buhr, "The Logical Design of Multiple Microprocessor Systems," Prentice-Hall Inc., Englewood Cliffs N.J., 1980.

Table 1: Definition of the Data Sets

C:	The list of predicted conflicts
L1:	The Short List These are resolution tactics preferred by the Controller, the selection of which was based on a set of criteria.
L2:	The short list with induced conflicts removed.
L3:	The set of resolution tactics acceptable under relaxed constraints. Note that L3 does not include the set L1, since L1 is known to induce conflicts.
L4:	The set of resolution tactics acceptable under relaxed constraints with all induced conflicts removed.
A:	Advice to the Controller including conflict data plus a prioritized list of resolution maneuvers.
HELP:	Conflict data, plus specifics on what has already been tried.

```
Function: Resolve_Conflict

;* Assume N = number of tactics to be examined
;*        L = number of induced conflicts
;* The algorithm in Figures 8 and 9 is executed using the global procedure names shown

BEGIN
  IF Sparse_Space() = OK THEN
  BEGIN
     Prioritize_and_compute();
  END
  ELSE
  BEGIN
    IF(Relax_and_Delete() = Null)OR (Detect_and_Remove = NULL)  THEN
    BEGIN
      IF (System = Autonomous) THEN   HELP();
      ELSE
      BEGIN
        IF (Partition_and_Detect() = NULL) THEN
        BEGIN
          CLASSIFY();
          Prioritize_and_Compute();
          Advise_Controller();
        END
        ELSE
        BEGIN
          FOR i = 1 TO N DO
            BEGIN
;* Compute the airspace state after the execution of the ith  tactic
              Compute Airspace [i]
;* compute the number of induced conflicts
              Detect Conflicts [L]
              F = True
              Count = 0
;* This determines if the number of conflicts is growing
              Number = Resolve Conflict
              IF Number NE 0 THEN
              BEGIN
                IF Number > = L THEN
                BEGIN
                  F =  False
                  i = L
                END
                Count = Count + Number
                IF Count > L THEN
                BEGIN
                  F = False
                  i = L
                END
              END
            END
;* The following two IFs either call for help or determine that a resolution has been found
            IF i = N AND F = False THEN
               NOT Resolving, Call for HELP
            IF L < N AND F = TRUE
               Resolved.
          END
        END
      END
      ELSE
      BEGIN
        CLASSIFY();
        Prioritize_and_Compute();
        Advise_Controller();
      END
   END
 END
```

Table 2: Pseudo Code for Autonomous Resolution

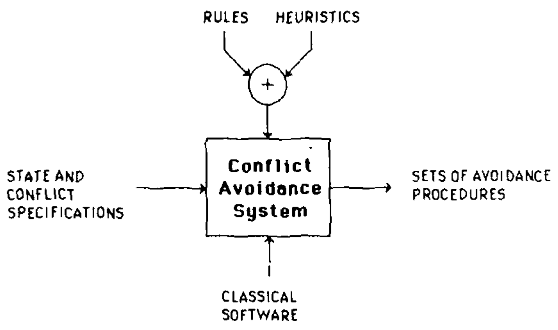

RULES
HEURISTICS
+
Conflict Avoidance System
STATE AND CONFLICT SPECIFICATIONS
SETS OF AVOIDANCE PROCEDURES
CLASSICAL SOFTWARE

Figure 1 A conflict Avoidance "Black Box"

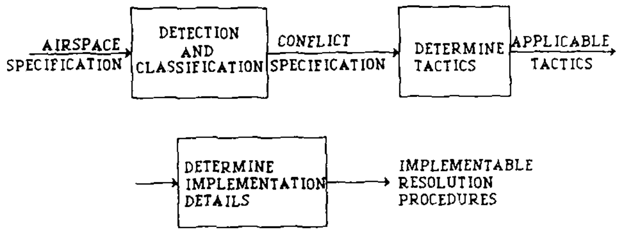

AIRSPACE SPECIFICATION
DETECTION AND CLASSIFICATION
CONFLICT SPECIFICATION
DETERMINE TACTICS
APPLICABLE TACTICS
DETERMINE IMPLEMENTATION DETAILS
IMPLEMENTABLE RESOLUTION PROCEDURES

Figure 2 Three Stages in Conflict Avoidance

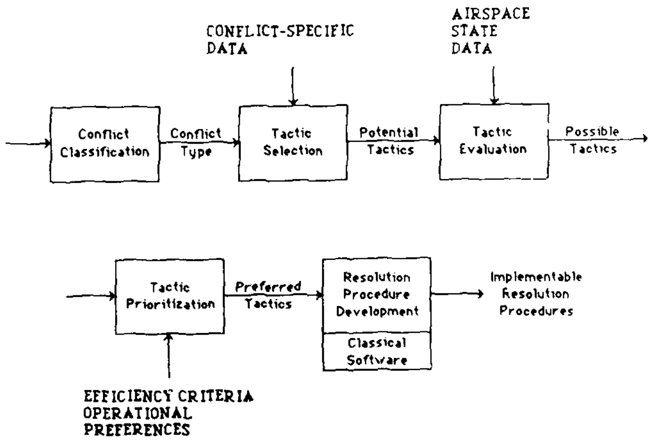


Figure 3 Global Inferencing algorithm

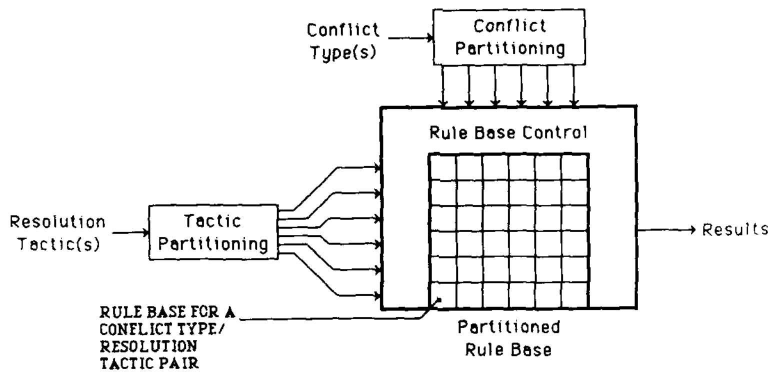


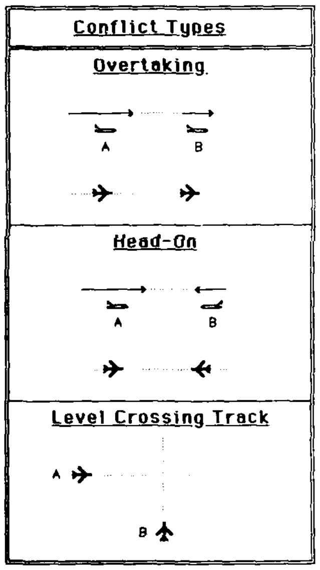
Figure 4 Rule Base Structure

Tactic Look-Up Table Subset

Table Entries

X - not applicable
A - applies to A only
B - applies to B only
o - applies to A or B

Tactics:		Overtaking	Head-On	Level Crossing Track
Change Flight Level	Up	o	o	o
	Down	o	o	o
Change Airspeed	Increase	B	X	o
	Decrease	A	X	o
Change Track	Port	o	o	B
	Starboard	o	o	A
Holding Pattern	Port	o	o	o
	Starboard	o	o	o



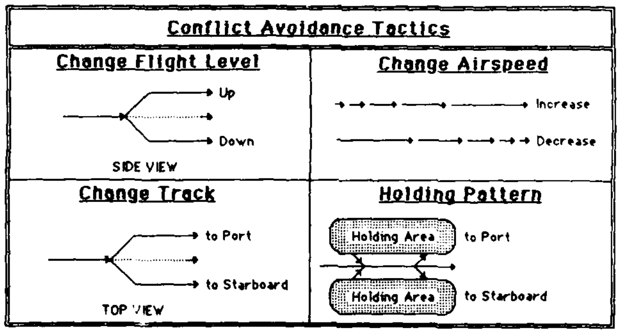


Figure 5 Tactic look-up Table

Test # 4	Conflict Type Head-On			
	Aircraft A	Aircraft B		
ID	Cessna	C5A Galaxy	Track Above	not occ
Airspeed	170	400	Track Below	not occ
Flight Level	150	150	Track Ahead	not occ
Attitude	level	level	Track to Port of A	occ
Next Maneuver	nothing	nothing	Track to Starboard of A	occ
Priority	avg.	avg.		
Pre-Conflict Angle	180		Weather Conditions	OK
Post-Conflict Angle	180		Restricted Airspace	to port

Figure 6 Sample Validation Test

Results:

	Tactics		Selection	Evaluation	Prioritization
1	Change Flight Level Up - A	✓	pass	pass	0.900
2	Change Flight Level Up - B	✓	pass	pass	0.800
3	Change Flight Level Down - A	x	x	x	x
4	Change Flight Level Down - B	x	x	x	x
5	Change Airspeed, Increase - B	x	x	x	x
6	Change Track to Port - A	✓	fail	x	x
7	Change Track to Port - B	✓	pass	pass	0.800

Comments:

1) Unconstrained case - results depend only on aircraft characteristics.
2) Tactic 6 failed selection because it's navigational ability is low, and is unable to navigate this turn.
3) The priority of tactic 1 is reduced because aircraft A (the Cessna) is above it's optimum altitude.
4) The priority of tactic 2 is reduced because the maneuverability of the C5A Galaxy is "low".
5) The priority of tactic 7 is reduced because the maneuverability of the C5A Galaxy is "low".

Figure 7 Results of Sample Test

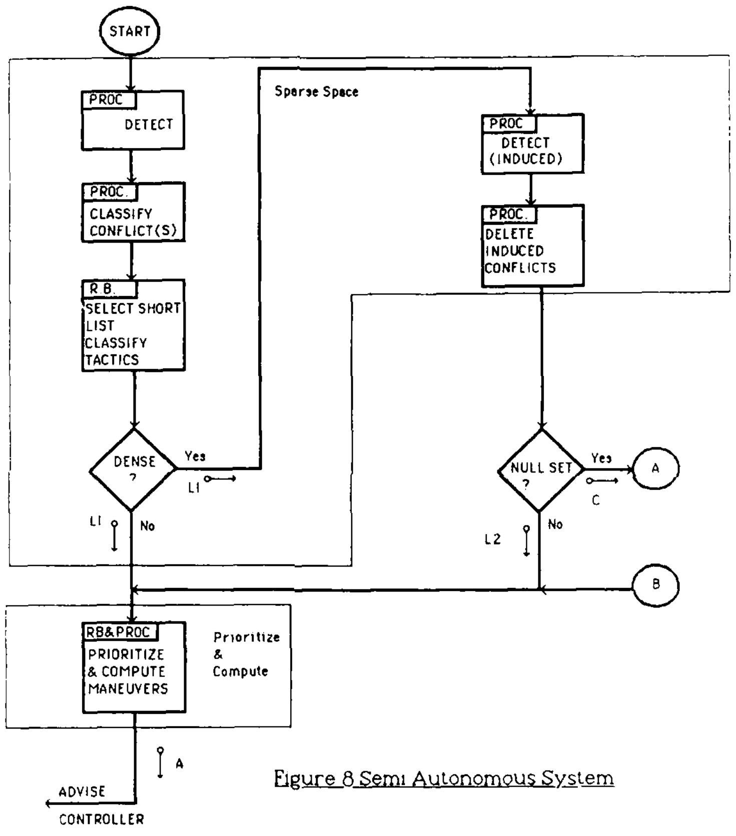


Figure 8 Semi Autonomous System

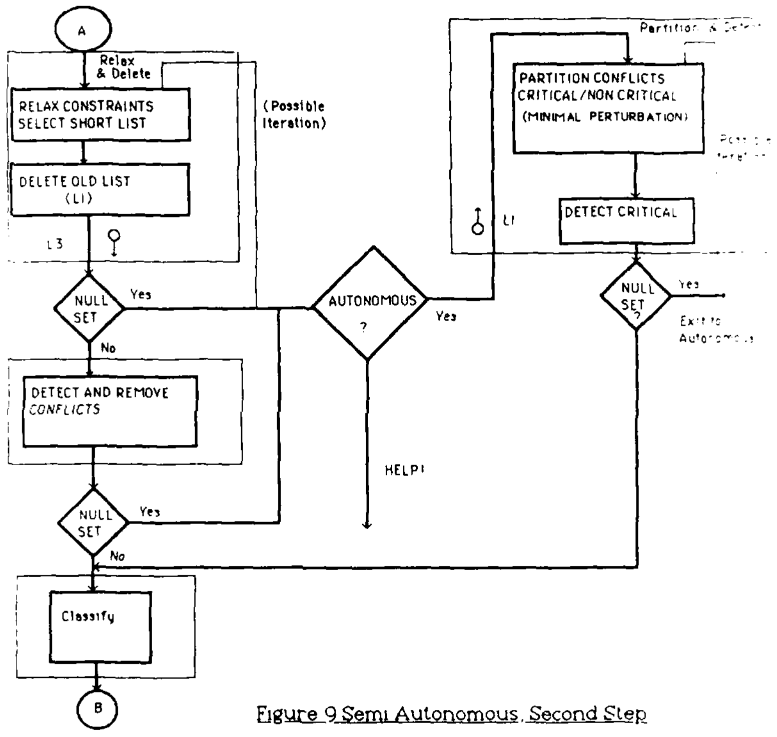


Figure 9 Semi Autonomous, Second Step

APPLICATION OF KNOWLEDGE-BASED TECHNIQUES TO AIRCRAFT TRAJECTORY GENERATION AND CONTROL

by
MICHAEL W. BIRD, PhD
Engineering Fellow
Lear Siegler, Inc./Instrument and Avionic Systems Division
4141 Eastern Avenue, S.E.
Grand Rapids, Michigan 49518-8727 USA

SUMMARY

A concept that embeds knowledge-based techniques in a trajectory generation and control system is defined. The system concept is called the Unified Trajectory Control System (UTCS). The objective of this system is to aid the pilot when operating in the intense threat environment projected for the 1990's.

The UTCS has an architecture of independent trajectory generation elements whose operations are integrated by a knowledge-based system. This Artificial Intelligence (AI) technique utilizes production rules, an inference engine, and a system of frames for communicating with the trajectory generation elements. Separate trajectory generation elements are utilized for terrain following/terrain avoidance, threat avoidance, weapon delivery, obstacle avoidance, and mission planning. The role of the knowledge-based system within the UTCS is defined and the various components of the system are described including the structure of production rules, the inference engine mechanization, the types of problem-solving knowledge needed in the rule base, and the frame system architecture. The various uncertainties in the UTCS are described and the need for a method to account for these uncertainties as part of the knowledge-based techniques is identified. A simulation of the system developed using the LISP and FORTRAN computer languages on a VAX 8600 computer is described.

BACKGROUND

The missions of the 1990's will place requirements on pilots and crews not seen in the past. Our aircraft operators will be facing ground-based and airborne defensive systems that are sophisticated, extremely lethal, and operating cooperatively to increase their effectiveness. This threat environment will be dense, mobile, and unpredictable, thereby increasing the number of unexpected events occurring during the missions. These events will include encountering unexpected threats, detecting unknown obstacles when flying at low altitudes, incurring damage to the aircraft and systems caused by the threats, and changing mission directives to accommodate the dynamics of the battle.

The pilot has always had to handle unexpected and unplanned events during the mission. The difference in the future is these events will be happening at a faster rate because of the higher threat density. This compresses the amount of time the pilot has to make decisions. These decisions will be more difficult because of the large number of threats plus the pressure to complete each mission in a fast paced war. The decision making problem faced by the pilot and crew is characterized as follows: 1) time compression - events happening rapidly and demanding quick decisions and control, 2) lots of surprises - events that were not anticipated during mission planning and must be handled in real-time, and 3) conflicting goals - situations where goals associated with the mission, survivability, and human factors conflict with each other.

One way to increase the pilot's and crew's effectiveness in these projected military environments is to provide real-time, comprehensive, and accurate information to the cockpit - information depicting the situation outside and inside his aircraft. Sensors, communication systems, diagnostic systems, and cockpit interface devices are being developed to provide information the pilot can use for assessing his situation. However, providing more information to the pilot or crew is not the complete solution for pilot decision aiding. The pilot will often not have sufficient time to adequately analyze the information and select effective actions. And, to complicate the problem, quick and accurate decisions will be required during critical situations involving rapid occurrences of dangerous events and requiring complex trade-offs. What is needed are computers that process the situation assessment information and derive data that aids the pilot in making decisions.

INTRODUCTION

A concept for pilot decision aiding is described in this paper. The concept has an on-board computer system computing one or more desirable aircraft trajectories for the pilot and controlling the aircraft to the trajectory selected by the pilot. The concept is called the Unified Trajectory Control System (UTCS). To be effective as a pilot decision aid, UTCS will need a number of capabilities:

- The trajectories will need to be computed to anticipate the approaching situation, where mission planning knowledge will be used, and to respond in real-time to unexpected events occurring in the actual situation. (To anticipate the approaching situation, mission planning knowledge will be used.)

- These trajectories must be computed with the fidelity needed for flight control and have the reliability necessary for flight safety.
- The trajectories need to avoid known threats, evade detected threats either before or after weapon launch, avoid unexpected obstacles during low-level flight, account for aircraft performance degraded by equipment failure or battle damage, and, when necessary, deviate from the mission plan in such a way that still accomplishes mission objectives as best as possible. For threat evasion, the UTCS needs to evaluate the use of aircraft maneuvers, terrain masking, countermeasures, weapons, or a combination of these for countering the threat.
- Preferences about the trajectory inserted by the pilot into the system need to be accommodated in the trajectory calculations.
- The various uncertainties and inaccuracies in the information utilized in the system must be accounted for in the UTCS calculations.
- In making trade-offs between conflicting goals, the UTCS's decision making logic must be compatible with the pilot's.

The complexities of the problem being solved by UTCS rule out a straightforward application of conventional trajectory optimization techniques. These techniques are numeric in nature, i.e., the problems being solved are modeled with variables having numbers as values and the problems are solved with equations and logic that operate primarily on numeric valued variables. The algorithms used in these techniques are derived principally from analytical methods including non-linear optimal control, singular perturbation theory, and dynamic programming. The current versions of these numeric-based algorithms have been applied to trajectory optimization problems involving a few performance criteria. However, when there are more than a few criteria, or trajectory goals, and when human expertise needs to be built into the system to handle conflicts between criteria, a single, analytically-based trajectory optimization algorithm is extremely difficult to develop.

The integration of knowledge-based, Artificial Intelligence techniques with a number of numeric-based trajectory optimization algorithms is the approach used in the UTCS concept. Artificial Intelligence concepts have been applied to a variety of challenging problems in recent years. From these applications has evolved a set of techniques for symbolically representing human expertise or knowledge. Symbolic representations have an advantage over purely analytical methods for representing human expertise. They are useful for heuristics or "rules of thumb" used to simplify the numerical calculation of solutions to problems.

Many Artificial Intelligence systems rely primarily on symbolic representations and processing. This is in contrast to UTCS which uses both symbolic and numeric processing. Symbolic representations do not have the fidelity needed for many of the trajectory computations. This is why numeric-based algorithms are utilized in UTCS. The UTCS concept uses a number of numerical trajectory generation algorithms with each algorithm computing trajectories for a few trajectory performance criteria. UTCS integrates these algorithms with the symbolically oriented Artificial Intelligence techniques.

The Artificial Intelligence techniques used in the UTCS concept have been successfully used in other AI applications. They are combined with reasonable, mature, trajectory optimization algorithms. This combination offers the potential of yielding a system that is realizable in the near future.

This paper describes the UTCS concept that has been developed during an Air Force Flight Dynamics Laboratory sponsored program [1]*. The description includes the following: a functional description of the concept, the numeric-based trajectory optimization algorithms, the Artificial Intelligence techniques employed in the concept, the type of knowledge needed for the system, and the progress made in simulating the concept. The paper concludes by describing what additional efforts are needed for further developing the concept.

UTCS FUNCTIONAL ARCHITECTURE

The UTCS has the two basic functions of a trajectory control system - trajectory generation and trajectory tracking. The trajectory generation function calculates the trajectory that is desirable for the aircraft to fly [2]. The trajectory tracking function computes commands for the flight control system and guidance cues for the pilot to follow the desired trajectory.

For trajectory generation, the UTCS approach employs a number of trajectory generation modules where each module computes a particular type of trajectory. Because each trajectory generation module has a specialty, the modules are referred to as trajectory specialists. The specific trajectory specialists are discussed below, but examples are terrain following/terrain avoidance, obstacle avoidance, and weapon delivery. Employing trajectory specialists allows the trajectory generation problem to be distributed among the specialists. Each specialist has its domain of expertise, a few trajectory optimization criteria, and its own optimization technique for computing optimum trajectories.

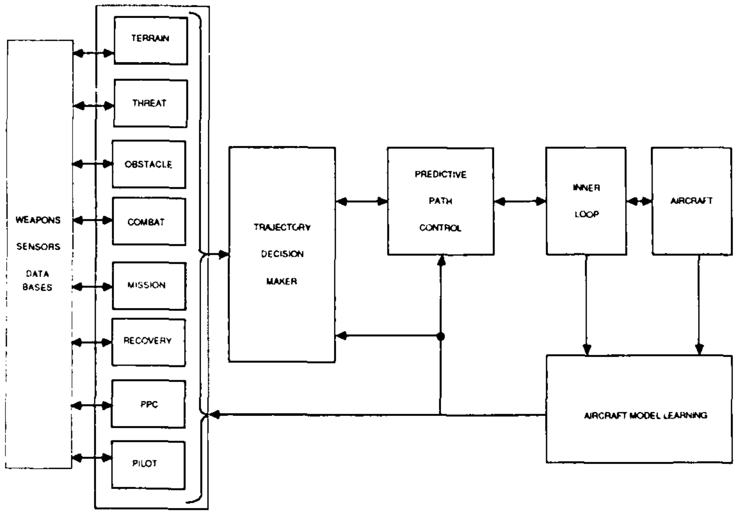
* Numbers in brackets designate references at end of paper.

With this approach, the total set of trajectory goals (criteria) are distributed among the various specialists. These specialists jointly contribute to the total solution of the trajectory problem.

In the UTCS architecture, the task of integrating the specialists' operations is handled by a separate module, the Trajectory Decision Maker (TDM). This module schedules the specialists' processing, makes trade-offs between conflicting trajectory criteria, and blends the individual trajectory specialists' outputs together into full trajectory solutions for the aircraft. Knowledge-based techniques are used for the TDM.

Figure 1 illustrates the UTCS functional architecture showing the trajectory specialists and the Trajectory Decision Maker. Shown in the figure is a direct interface between the specialists and the sensors, weapons, and data bases that the specialists need to accomplish their particular task. Since the TDM is involved in the higher level decision making aspect of the problem, it does not need the same level of information detail from the sensors, weapons, and data bases as the specialists. Therefore, an interface to the TDM is not shown on the architecture diagram.

Figure 1 also shows the trajectory tracking function of the UTCS, which is comprised of the Predictive Path Control and inner loop control laws. The TDM feeds the desired trajectory to Predictive Path Control for the tracking operation. Predictive Path Control is an advanced control technique that has been developed as part of the UTCS program. A feature of this control technique is accurate trajectory tracking of the dynamics in the commanded trajectories. This is accomplished in this technique by continually predicting the best set of controls for the immediate trajectory future [2].



UTCS FUNCTIONAL PARTITIONING

FIGURE 1

Another element of the UTCS concept shown in Figure 1 is Aircraft Model Learning. The other UTCS elements - the trajectory specialists, the TDM, and the Predictive Path Control - require accurate information about the current aircraft aerodynamic, propulsion, and control capabilities. This is the role of Aircraft Model Learning. This UTCS element will update models and any other types of representations of aircraft capabilities that are utilized in the other UTCS elements. The updating is especially critical when on-board subsystems degrade in performance or fail and when there is damage to the aircraft. The module will utilize data from sensors, predicted aircraft trajectories, and subsystems BIT (Built-In Test) and diagnostic systems. A combination of system identification and performance estimation techniques will be used in Aircraft Model Learning for processing this data.

Knowledge-based techniques will be used in all elements of UTCS, not just in the Trajectory Decision Maker, to provide the flexibility needed for the aircraft to operate in the hostile environment described above. However, the initial development of the UTCS concept emphasized applying knowledge-based techniques to the TDM and its integration with the trajectory specialists. Consequently, the remainder of this paper is devoted to describing the TDM, the trajectory specialists, and their combined operation.

TRAJECTORY SPECIALISTS

Low altitude flight in the presence of threats is most demanding in terms of the rate of unexpected events and, consequently, the needs of a trajectory control system for pilot aiding. This type of mission segment was used to develop the UTCS concept. The trajectory specialists selected for this mission are shown in Figure 1. Their functions are as follows:

1. Terrain for low altitude trajectories optimized with respect to the terrain.
2. Threat for trajectories and countermeasure usage that maximize survivability against threats affecting the aircraft's current operation.
3. Obstacle for trajectories avoiding unexpected obstacles.
4. Combat for delivering weapons against targets or threats.
5. Mission for in-flight planning of new mission trajectories. This involves computation of long term trajectories that meet mission objectives while maximizing survivability against anticipated threats.
6. Recovery for recapturing the current planned mission when deviations from the mission plan occur.
7. Predictive Path Control (PPC) for predicting the aircraft trajectory that will result when the TDM supplies a desired trajectory to the trajectory tracking function. The PPC was described earlier as a part of the trajectory tracking function, but it is also employed as a specialist in the UTCS concept. The predicted trajectory the PPC computes when in the specialist role accounts for the control actions of the PPC (when operating as a controller), the effects of the inner loop control laws, the aircraft aerodynamics, and the propulsion system.
8. Pilot for incorporating the pilot's trajectory preferences and constraints. This specialist keeps track of the trajectory preferences the pilot makes as the flight progresses and supplies these pilot desires to the TDM. This specialist is not a model of the pilot and is not an attempt to emulate his cognitive process, but is a mechanism that feeds pilot preferences or demands about the trajectory into the trajectory decision making process. These preferences are expressed by the pilot through the control display interface.

The UTCS function is generating the desired near-term trajectory for the aircraft. A near-term trajectory originates from the aircraft current state and is defined over a reasonable length of time in the future. Consequently, all trajectory specialists, except the mission specialist, compute near-term trajectories and are concerned with satisfying the trajectory goals for the local situation effecting the aircraft. However, UTCS must also consider how the near-term trajectory affects the future parts of the mission. This is the role the mission specialist plays.

Predictive Path Control has a unique role as a specialist. When the TDM gives the PPC a potential desired trajectory computed by the other specialists, the PPC feeds back to the TDM the effects the control system and aircraft dynamics will have on the trajectory. This allows the TDM to evaluate the effects of control actions before committing to a desired trajectory. This capability is important for the UTCS to meet flight safety standards.

Many of the trajectory generation algorithms that can be used in the trajectory specialists have already been developed. Terrain following/terrain avoidance algorithms [5] can be used for the Terrain specialist and weapon delivery algorithms [6] for the Combat specialist. Route planning algorithms have been recently developed for Mission Planning [2]. Of all the specialists, the least developed specialist is Threat. This specialist computes the near-term trajectory and recommended countermeasure usage to avoid known threats while evading detected threats.

TRAJECTORY DECISION MAKING AND TRAJECTORY SPECIALISTS' OPERATIONS

The trajectory specialists compute their own individual trajectories, while the TDM manages the specialists' computations to establish complete trajectories that best satisfy all trajectory goals. In managing the specialists, the TDM performs a number of operations. The TDM decides which specialists should contribute to the desired trajectory, based on events occurring both external and internal to the aircraft. It combines the individual outputs of the trajectory specialists into composite candidate trajectories that attempt to satisfy all factors affecting the aircraft trajectory. The TDM analyzes these candidates to establish if any satisfy all the trajectory factors, if conflicts between factors need to be resolved, and if trajectory improvements are required. To resolve conflicts or improve the candidates, the TDM reschedules the specialists with different weightings on the specialists' optimization criteria and/or different constraints on the specialists' solutions. Whenever the specialists generate their individual trajectory solutions, new composite candidates are created and the TDM analyzes the results to determine the next actions in the process of determining a desired trajectory. This process of searching for desired trajectory solutions is an iterative process involving one or more specialists at each step and managed by the TDM.

When the TDM has determined one or more candidate trajectories that best satisfy all trajectory goals, the desired trajectory that will be input to the trajectory tracking function must be selected from these candidates. The method of selection will depend on how the UTCS is integrated with the pilot's operations in the cockpit. In one method, UTCS provides the most promising candidate trajectories to the pilot, via displays, who selects the desired trajectory. Another method, which may be used in time critical, high workload situations, has the TDM select the best candidate and automatically feeding it to the trajectory tracking function with minimal or no pilot involvement. It is anticipated that the selection method will vary during the mission and depend on what events are occurring and the level of pilot workload.

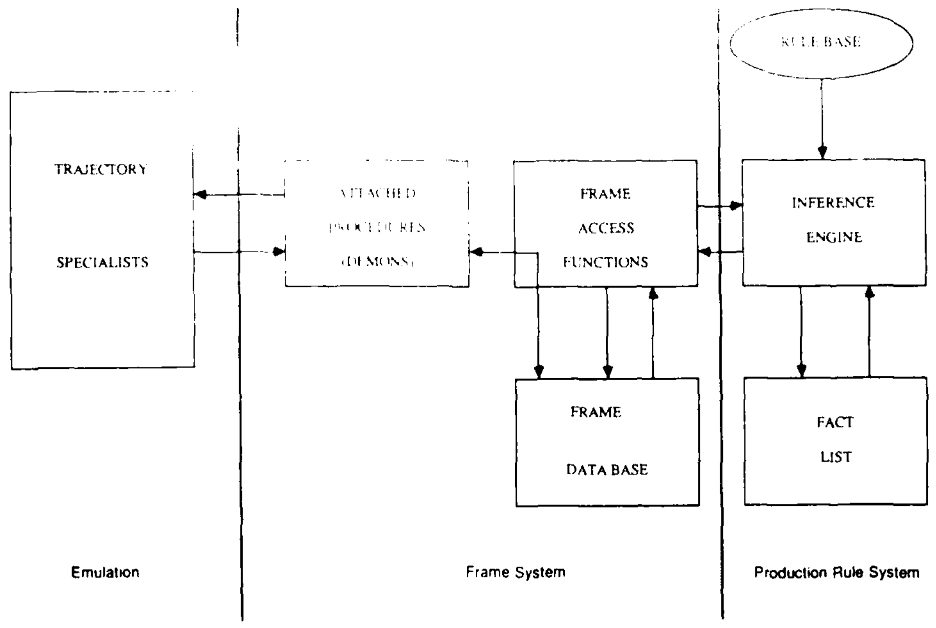
Computing aircraft trajectories with independent trajectory specialists whose operations are integrated and managed by the TDM is the trajectory generation concept of the UTCS. When addressing the development of this concept for real-time, airborne application, there is concern that the iterative search process described above will require unrealistically large computer throughput. Another concern is that there will be considerable inefficiency in the translation between the trajectory specialists' numerical representations and the symbolic representations of the Artificial Intelligence techniques. To address these concerns, two features were incorporated in the TDM specialists' concept.

One feature is having the specialists generate coarse, low resolution, abstract representations of trajectories when the TDM is exploring promising trajectory candidates. This allows the TDM to quickly examine many candidate solutions. When promising candidates have been established, the TDM has the specialists refine the candidates with fine, detailed calculations of the trajectories. This feature requires that the specialists compute both coarse, macroscopic representations and fine, detailed representations of their trajectory solutions. Furthermore, the specialists will need to generate the coarse, low resolution trajectories significantly faster, and with less computer speed requirements, than the fine, detailed trajectories.

The second feature of the integration approach is a language for representing aircraft trajectories. The language is needed to provide a trajectory representation common to all specialists, plus a means for them to communicate efficiently with the TDM. This language needs to be symbolic for the coarse trajectories and numeric for the detailed trajectories. The symbolic language provides for efficient processing of the trajectory information by the TDM's Artificial Intelligence techniques where knowledge about significant trajectory attributes, trade-offs, and the specialists' characteristics is defined symbolically. The numeric representation provides the accuracy needed for factors such as aircraft dynamics, terrain constraints, and threat coverage.

TRAJECTORY DECISION MAKER

Figure 2 shows the major modules of the Trajectory Decision Maker and the two Artificial Intelligence techniques used in the TDM. These two techniques are a production rule system and a frame system. The frame system interfaces the production rule system with the trajectory specialists.



TRAJECTORY DECISION MAKER

FIGURE 2

As shown in Figure 2, the production rule system has a knowledge base, a set of rules, a list of facts, and an inference engine. The fact list contains information about the local situation and facts describing the current state of the TDM trajectory search process. The inference engine matches the facts and trajectory information from the frame system to the set of rules and selects a rule to activate or "fire". A rule firing can have a number of results: a fact or facts can be added to or updated in the fact list, objects or data in the frames can be modified, new frames can be instantiated, and trajectory specialists' operations can be initiated. The inference engine processing is iterative: a match against the rule base, a rule firing, another rule base match now with the revised fact list and/or frames, another rule firing, and this continues until a rule firing terminates the process. It is critical that a computationally efficient procedure be used for this processing. Note: the simulation of the TDM engine used a forward chaining approach for processing the rules.

There is a distinction between the production rule system used in the TDM and the typical production rule system which involves just a rule base and fact list. For the TDM, information from the frame system is also processed in conjunction with the fact list. This difference results in a specialized structure for the rules that includes both facts and frame data.

The general IF-THEN rule structure that has been utilized in the knowledge bases of many production rule systems [3] was used for developing and simulating the TDM concept. However, since the TDM rules are expressed in terms of both symbolic facts and symbolic and numerical trajectory descriptions from the frame system, an IF-THEN rule structure was specifically developed for the TDM.

The IF part of each rule consists of a list of pattern clauses and a list of test clauses. The pattern clauses are symbolic expressions involving the facts in the fact list and can include pattern variables. The test clauses involve primarily the symbolic and numeric data from the frames and are composed of frame access functions for extracting information from the frames and auxiliary functions for processing the fact list. Including test clauses in the IF part of the rules was necessary for the TDM application so the rules could be expressed in terms of the trajectory specialists' outputs without having to convert the specialists' outputs into facts for the fact list.

The THEN part of a rule defines actions to be taken when a rule is fired and consists of functions for manipulating the facts in the fact list, data in the frames, and operations of the inference engine. In addition to the IF-THEN parts, a third part of the TDM rule structure is a rule confidence value, i.e., a numerical parameter defining confidence in the rule. Attaching confidence values to rules was part of a method used when simulating the TDM to account for uncertainties in the information processed by the TDM.

A part of the inference engine that significantly affects the behavior of a production rule system and, therefore, the behavior of the TDM is the conflict resolution strategy. This is the strategy the inference engine uses for selecting the rule to fire from the set of rules whose IF conditions match the fact list and frame data. In developing the TDM concept, a number of needs was established for the conflict resolution strategy.

First, the conflict resolution strategy must be changeable for different TDM operations. For example, the operation of deciding which trajectory specialists to employ needs a conflict resolution strategy different than for the operation of evaluating the desirability of each candidate trajectory. To be able to change the conflict resolution strategy implies that the THEN part of a rule should be able to specify the strategy being used by the inference engine.

Second, the methods for conflict resolution must include the processing needed to make decisions in the presence of uncertainty. The types of uncertainties and inaccuracies present in the UTCS that will influence the trajectory decisions are described in a later section. These uncertainties and inaccuracies need to be accounted for in the process of selecting the rules to be fired.

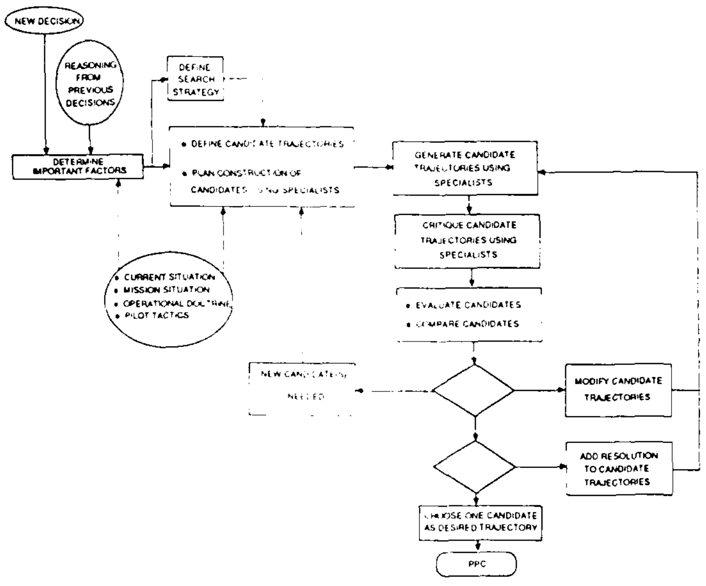
When simulating the TDM inference engine during the UTCS project, a technique based on the certainty factor propagation method developed as part of the MYCIN medical diagnosis research work [4,7] was used to account for uncertainty. In this technique, each fact in the fact list has a certainty factor which is a number depicting its certainty. The inference engine uses the certainty factors attached to facts for computing a certainty value for each rule whose IF conditions match the fact list and frame data. The inference engine computes this rule certainty based on the certainty factors of the facts in the rule IF condition and the rule confidence value. These rule certainty values can be used in conflict resolution methods for determining which rule to fire. Two conflict resolution methods based on rule certainty values were used in the TDM simulation. One method fired the rule with maximum certainty, while the other method prevented from firing any rule below a certainty threshold. The last step of the certainty factor technique occurs when a rule is fired. The certainty factor of each fact in the THEN part of the rule is computed based on the rule certainty and the previous value of the fact.

Third, the conflict resolution strategy needs to be mechanized as a hierarchy of conflict resolution methods. In the strategy, each method is assigned a priority. The inference engine applies the methods in order of priority with each method narrowing the set of potential rules for firing until one is fired.

TDM KNOWLEDGE BASE

The TDM rule base will have the knowledge needed for managing the trajectory specialists and using the specialists' outputs to develop desirable trajectories. The rule base will have heuristics for the following: deciding how to respond to unexpected events, what specialists to schedule, when to schedule them and what inputs to provide to them, evaluating the specialists' outputs, priorities on conflicting trajectory goals, trade-offs between trajectory goals as a function of the external and internal environment, and changes to make to the specialists' inputs to improve a candidate trajectory's desirability. The expertise needed for developing this TDM knowledge base will need to be derived from a combination of pilots, military mission analysts, and engineers with intimate knowledge of the trajectory specialists' algorithms.

The UTCS project was an exploratory development program, so resources were not available to develop a complete knowledge base. However, a prototype rule base was developed and simulated. To assist in developing this rule base, a basic approach for solving the TDM problem was established. Figure 3 defines the steps in this approach. These steps are described next.


NEW DECISION
REASONING FROM PREVIOUS DECISIONS
DETERMINE IMPORTANT FACTORS
DEFINE SEARCH STRATEGY
• DEFINE CANDIDATE TRAJECTORIES
• PLAN CONSTRUCTION OF CANDIDATES USING SPECIALISTS
• CURRENT SITUATION
• MISSION SITUATION
• OPERATIONAL DOCTRINE
• PILOT TACTICS
GENERATE CANDIDATE TRAJECTORIES USING SPECIALISTS
CRITIQUE CANDIDATE TRAJECTORIES USING SPECIALISTS
• EVALUATE CANDIDATES
• COMPARE CANDIDATES
NEW CANDIDATE(S) NEEDED
MODIFY CANDIDATE TRAJECTORIES
ADD RESOLUTION TO CANDIDATE TRAJECTORIES
CHOOSE ONE CANDIDATE AS DESIRED TRAJECTORY
PPC

PROBLEM SOLVING APPROACH

FIGURE 3

Determine Important Factors. This first step starts the problem solution, occurring whenever a new trajectory computation is needed. The UTCS as a real-time, on-board system will be making trajectory computations as the aircraft flight progresses, both periodically and whenever unexpected events occur. Many of the trajectory computations will be updates to the previously selected desired trajectory. However, when unanticipated changes occur in the current situation, the trajectory computations likely will result in recommendations for or selection of a new desired trajectory. This step in the problem solving approach determines what factors are important in the current trajectory computation. Examples of important factors are new events, such as a newly detected threat or temporary loss of a control element, and past trajectory decisions, such as a decision to attack an air threat or type of obstacle avoidance maneuver.

Define Search Strategy. Based on the important factors, the TDM establishes a strategy for determining candidate trajectories during the computational cycle. There are at least three strategies: 1) breadth-first, which is used when unexpected events occur dictating a search for new desired trajectories that may differ significantly from the current trajectory, 2) depth-first, which is used when there have been minor changes in the situation, i.e., updated threat information or minor aircraft deviations from the desired trajectory, requiring only refinements to the current desired trajectory, and 3) trajectory extension, which is used when there have not been any significant changes to the situation and the desired trajectory needs only to be extended because of aircraft progress along the path.

Define Initial Candidate Trajectories. The TDM establishes which specialists to employ for generating an initial set of candidate trajectories. A candidate can be a trajectory from a single specialist or a composite trajectory formed by trajectories from different specialists. Characteristics of the current situation, new events that have occurred, decisions made in past computation cycles, and the search strategy determine what specialists' combinations have the potential of generating candidate trajectories that best meet all trajectory goals. To generate different candidate trajectories, the TDM can adjust the specialists' inputs, which are the weightings on their performance criteria, trajectory constraints, and size of the trajectory search space.

Generate Candidate Trajectories Using the Specialists. The TDM activates the specialists or sequence of specialists to compute the candidate trajectories. Each specialist computes the trajectory that is optimum relative to its domain of expertise.

Critique Candidate Trajectories Using the Specialists. The TDM has each specialist evaluate the performance of all candidate trajectories relative to their domain of expertise. The array of critiques provided by the specialists are used by the TDM in the next step to evaluate the performance of each candidate relative to all trajectory goals.

Evaluate and Compare Candidates. The performance of the candidate trajectories against the trajectory goals is evaluated and, when necessary, the performance of different candidates are compared. These evaluations and comparisons can have four types of trajectory search actions: 1) modifications to existing candidate trajectories to improve their overall performance by changing constraints to the specialists' optimization problems, 2) generation of entirely new candidates by using specialists or combinations of specialists that were not previously used, 3) refinement of coarse, low resolution candidates that are considered desirable by a return to the specialists for a detailed trajectory calculation, and 4) acceptance of one or more candidates as desirable trajectories. If any one of the first three actions occurs after an evaluation, the specialists are activated to compute the candidates and the evaluation/compare step is performed again. The iteration through the evaluation/compare step continues until one or more trajectories are found desirable. When candidates cannot be found that satisfy all trajectory goals, priorities on the goals and/or goal thresholds are modified until a successful candidate (or candidates) is developed.

THE FRAME SYSTEM

As illustrated in Figure 2, the UTCS concept uses a system of frames as the communication medium between the TDM and the trajectory specialists. A frame is defined as a data structure describing a class of objects and consists of a collection of slots that describe the various aspects of the object. The value of a slot can be another frame, a frame feature which gives rise to a hierarchy of frames [4].

Frames are used for a number of purposes in UTCS as shown in the frame hierarchy of Figure 4. The four sets of frames shown in the figure represent the following:

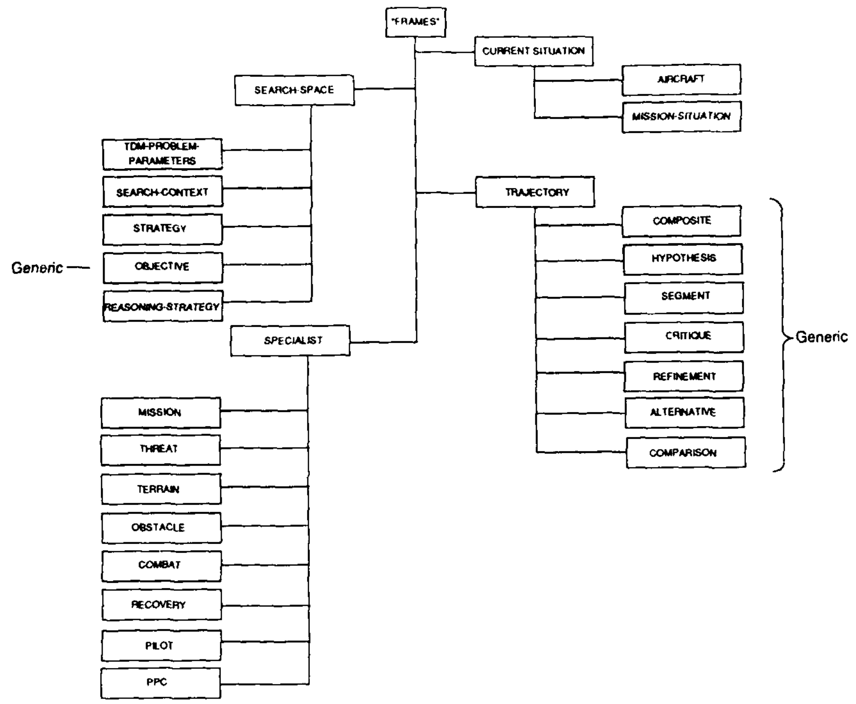
Current Situation represents the aircraft state and the mission situation, such as threat status and mission critical points.

Search Space represents items such as the volume of the trajectory search space and the search strategy.

Trajectory is a hierarchy of frames representing the trajectories. At the top of this hierarchy are the composites, the TDM candidate trajectories. Composites are a combination of hypotheses which are the trajectories generated by the individual specialists. The hypotheses are divided into segments by the specialists as a means for representing the different aspects of the trajectories. Other frames associated with trajectories represent critiques, refinements, trajectory alternatives, and comparisons between composites.

Specialist contains the procedures for inputting data and receiving data from the different specialists.

The frame mechanism has two features - generic frames and attached procedures - that have been useful in UTCS. Generic frames have been employed extensively for storing the specialists' trajectories where frames for composites, hypotheses, and segments are instantiated whenever a specialist is activated. Attached procedures, often called demons, have been an efficient way for the TDM inference engine to invoke a specialist. Whenever a rule firing calls for a particular specialist to be processed, a value of a particular slot in the specialist's frame is set. This initiates a procedure that activates the specialist. The inference engine is not involved in setting up the specialist's inputs or receiving the specialist's inputs.



TRAJECTORY DECISION MAKER FRAME SYSTEM

FIGURE 4

In summary, the combination of a frame system and a production rule system is well suited to the UTCS. The use of generic frames and a hierarchy frame structure is efficient for representing trajectories. Activating the specialists as attached procedures in the frame slots eliminates the need for those detailed procedures being mechanized in the production rule system. Storing the trajectory data in frames instead of as facts in the fact list drastically reduces the size of the fact list and, correspondingly, the execution time of the production rule system.

TRAJECTORY SPECIALIST PROCESSING

In the UTCS concept, each specialist has its own algorithm and, perhaps, Production Rule System for computing a trajectory that is optimum or desirable for its domain. However, in interfacing with the frame system, all the specialists will have a number of common elements. Figure 5 illustrates a general structure for each specialist's processing. The frame system provides a symbolic representation of specialists' inputs which are converted into a numerical representation of the inputs using a decoder function. The set of inputs is then transformed into an appropriate form for the specialist trajectory processing.

Each specialist will have two modes: trajectory generation and trajectory critique. The critique mode provides data to the TDM which it uses to evaluate the other specialists' trajectories in terms of this specialist's domain and objectives. For trajectory generation, the specialist computes the trajectory optimum relative to its domain, using models, data bases, knowledge bases, and optimization techniques appropriate to the processing task. After being generated, the trajectory's performance is critiqued relative to the specialist's objectives and criteria. This is the same critique process that is performed when the TDM requests the critique of an input trajectory. After the critiquing is complete, a trajectory encoder is used to transform the trajectory and its critiques into the symbolic representation which is output to the frame system.

The TDM provides the same types of inputs to each specialist. These inputs include the following: 1) a 3-D trajectory search space, including a starting aircraft state and optionally a terminal state, 2) a priority ranking of the optimization criteria used by the specialist, 3) a generate/critique request, 4) a representation resolution request (coarse, intermediate, or fine), and, optionally, 5) a reference trajectory with tolerance band. If this latter option is selected by the TDM, the specialist constrains its trajectory optimization search to within the tolerance band around the reference.

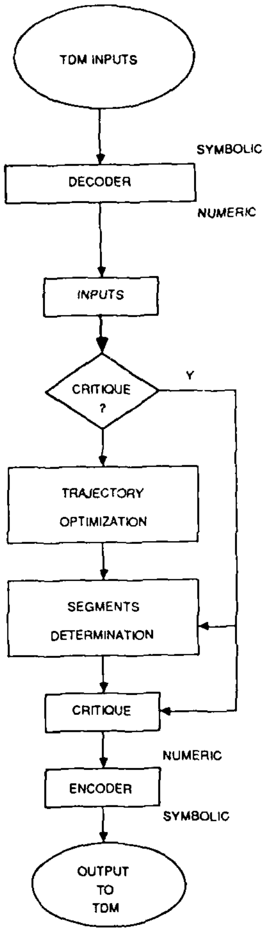
The general structure for the specialists shown in Figure 5 has been designed to accommodate existing trajectory generation algorithms, such as terrain following/terrain avoidance [5], maneuvering weapon delivery [6], and Tactical Flight Management global trajectory generation [2]. These algorithms will be used in the trajectory generation block of the general structure. Existing trajectory generation algorithms will typically be computing trajectories with only one resolution; therefore, for the UTCS application, these algortihms will need to be expanded to compute trajectories with coarse, intermediate, and fine resolutions.

DECISION MAKING IN PRESENCE OF UNCERTAINTY

As indicated earlier in this paper, the TDM needs to account for the different levels of uncertainty and inaccuracy inherent in the data and rules that the UTCS uses to make trajectory decisions. Uncertainties and inaccuracies exist in various forms in this system.

Inaccuracies are expected in data provided by sensors. For the UTCS, this includes data describing ground threats, air threats, obstacles, and aircraft and control dynamics. These inaccuracies are of different types: detection inaccuracies, classification imperfections, and parameter errors.

The UTCS also exchanges information with other on-board systems, not involving sensing, and there will be uncertainties in some of the information provided by these systems. For example, key mission events (coordination points, time constraints, and targets) will have varying degrees of importance. This variation in importance is a form of uncertainty. Another example is pilot trajectory preferences that are input into the system via the cockpit controllers. There will be variations in the strength of pilot preferences, another form of uncertainty.



GENERAL FLOW DIAGRAM FOR EACH SEGMENT

FIGURE 5

In addition to uncertainties in sensor data and information, there will be uncertainties in the knowledge base. The set of rules making up the knowledge base will have varying levels of credibility. In the MYCIN certainty factor method, a confidence factor is assigned to each rule by the human expert creating the rule to quantify its credibility.

A method for accounting for these types of uncertainties is required in the UTCS concept. The method most applicable to UTCS needs to be extracted from the research that has been and is being conducted in the area of inexact reasoning [7], [8], [9]. During the UTCS project, a MYCIN-type certainty factor method was used in simulating the TDM inference engine, as described above. Working with this method, a concept for handling the types of uncertainties present in UTCS was formulated.

In the UTCS concept, some of the facts contained in the fact list symbolically describe such things as the elements of the current external situation, key parts of the mission plan, the pilot's trajectory preferences, and health of the aircraft and its systems. These are macroscopic descriptions appropriate to the decision making aspects of the TDM. In the certainty factors method, a certainty factor is associated with each fact in the fact list. Therefore, certainty factors can be used to characterize the reliability of the information contained in the facts, i.e., certainty factors characterize the reliability of the current situation information, the criticality of the mission points, the definiteness of the pilot's preferences, and the reliability of the on-board system health assessments. With these factors describing the certainties of the current and future situations, the inference engine propagates the factors through the rules and, combines these factors with the rule confidence value, and makes decisions from the resulting rules' certainty values. These decisions are such things as which trajectory specialists to invoke, what sensor information each specialist uses in generating its trajectory outputs, and how to weigh each specialist's outputs in terms of its reliability when making trade-offs between trajectory goals.

In developing the concept for handling uncertainty, the interplay between the TDM and the trajectory specialists was addressed. One approach to establishing the roles of the TDM and the specialists is to have each specialist account for the uncertainties in the sensor data and information it uses when computing its trajectories. With this approach, the specialists would provide their trajectories to the TDM along with indications of trajectory certainty. The disadvantage with this approach is the TDM would be unable to vary the level of assumed sensor performance or to select what sensor data to use when creating trajectory candidates with the specialists. For example, the TDM may want the terrain specialist to generate trajectories assuming, first, normal performance from the terrain sensors and, then, worst case performance from the sensors. Similarly, the TDM may want the threat specialist to generate trajectories for only threats known with high probability and also to generate other trajectories with the threat data base including known as well as suspected threats. For this reason, the recommended approach for dividing the responsibility between the TDM and specialists is to first establish what sensor data and sensor performance levels the TDM will need to control to create trajectory candidates especially when considering trade-offs between trajectory goals. Second, expand the interface between the TDM and specialists so the TDM can specify to each specialist, when it is activated, the sensor data to use and the level of sensor performance to assume when computing its trajectory.

UTCS SIMULATION

A simulation of the UTCS concept was performed during the UTCS Program on a VAX 8600 computer. A short flight segment of a low altitude mission was performed. The major parts of the concept were simulated - the TDM, five of the trajectory specialists (threat, terrain, mission, recovery, air-to-air combat), and the PPC. The simulation included models of a high fidelity fighter aircraft and associated inner-loop flight control system. The simulation used DMA (Defense Mapping Agency) terrain elevation data from the Fulda Gap area and including a number of ground threats. Generic lethality models were used for the ground threats. Detection of an unexpected ground threat and an air threat were the events simulated.

Two Higher Order Languages (HOLs) were used for the simulation software development: VAX LISP (the Digital Equipment Corporation Common LISP language) and FORTRAN. The trajectory specialists, the aircraft and control models, and the PPC were programmed in FORTRAN because they have involved primarily numerical calculations. The production rule system and frame system were programmed in VAX LISP for ease of developing the symbolic processing part of the concept.

Because the UTCS Program was an exploratory development program, only a prototype set of rules for the TDM was developed and implemented. A total of 140 rules were implemented to illustrate all operations of the TDM in the UTCS concept. A significantly larger knowledge base will be needed in a fully operational system. Also, because of the exploratory nature of the project, the trajectory specialists were not developed. Instead, their behavior was emulated using a dynamic programming algorithm and software code previously developed on the Air Force Tactical Flight Management Program [1].

CONCLUSION

The UTCS concept has been defined and a top-level simulation of its characteristics has been performed. The concept is a trajectory and attitude control system for pilot decision aiding designed for operation in an environment of unplanned events, changing missions, and damaged aircraft expected in the intense threat environment of the 1990's. A functional architecture of independent trajectory specialists integrated with a knowledge-based trajectory decision maker is the cornerstone of the concept. The issues involved in the integration of the numerical processing and symbolic processing that is inherent in this architecture and any knowledge-based trajectory control system have been addressed.

This concept was developed as part of an exploratory development program. Further development of the concept will need to include the following developments: the TDM knowledge base, the threat and mission trajectory specialists, techniques for handling uncertainty, the aircraft model learning module, a cockpit integration concept, and a computer architecture for parallel implementation of the specialists and TDM.

REFERENCES

1. Anon., "Safety-of-Flight Analysis and Design of Trajectory Control Systems", Lear Siegler, Inc./Astronics Division, Santa Monica, California, ADR 871, 1987.

2. M. D. Olinger and M. W. Bird, "Tactical Flight Management: Threat Penetration Algorithm Design", May 1984, IEEE NAECON, Dayton, Ohio.

3. F. Hayes-Roth, D. A. Waterman, and D. B. Lenat (eds.), Building Expert Systems, Addison-Wesley, Reading, MA, 1983.

4. E. Rich, Artificial Intelligence, McGraw-Hill, New York, 1983.

5. W. W. Harrington and T. G. Gates, "Terrain Following/Terrain Avoidance (TF/TA) Design Evaluation", May 1985, AFWAL Paper.

6. R. J. Landy, R. P. Meyer, R. E. Lampert, and D. C. Pruess, "Integrated Flight and Fire Control: System Integration and Testing and the Role of On-Board Simulation", AIAA-82-1625, August 1982, AIAA Guidance and Control Conference.

7. E. H. Shortliffe and B. G. Buchanan, "A Model of Inexact Reasoning in Medicine", Mathematical Biosciences, Vol. 23, 1975, pp. 351-379.

8. L. A. Zadeh, "Fuzzy Sets as a Basis for a Theory of Possibility", Fuzzy Sets and Systems, 1978.

9. G. Shafer, A Mathematical Theory of Evidence, Princeton University Press, Princeton, N.J., 1976.

ACKNOWLEDGEMENTS

This work was supported by the AFWAL Flight Dynamics Laboratory under Contract No. F33615-84-C-3626. Charles Wagner served as AFWAL Project Engineer and Genese L. Fileccia as Group Leader. The Astronics Division of Lear Siegler, Inc. was prime contractor for this program with the Instrument and Avionic Systems Division of Lear Siegler as subcontractor.

The author would like to especially acknowledge the contribution of the late William Swartz of the AFWAL Flight Dynamics Laboratory. His advocacy, devotion, and insights made major contributions to the advancement of trajectory and attitude control technology.

TOWARDS THE UNMANNED COCKPIT

by
Dr Brian Ellis
Head Human Engineering Division
ROYAL AIRCRAFT ESTABLISHMENT
Farnborough
Hants
GU14 6TD
UK

"I must Create a System, or be enslav'd by another Mans"

William Blake, "Jerusalem".

SUMMARY

Trends in air-warfare make the development of autonomous unmanned aircraft necessary. Advances in Intelligent Knowledge Based Systems (IKBS) and in computing technology will make it possible. This paper examines the case for unmanned aircraft and their probable evolution from manned platforms with discrete intelligent aids through more sophisticated, highly interactive IKBS to eventual autonomy. Some indications are offered of the developments that will be called for in the IKBS themselves and in computing hardware and some of the problem areas that are already known, such as validation and knowledge elicitation are considered in some detail.

1 WHY CONSIDER AN UNMANNED COCKPIT?

There are two main driving forces that lead us to consider future air systems platforms which do not carry a human pilot but which perform many of the functions of today's piloted aircraft (helicopters as well as fixed wing). Firstly, there is the continuing escalation in the complexity and workload involved in the modern military aircraft mission, and, secondly, there is the awareness of the enormous strides that are being made in computer power and data storage, and which will undoubtedly continue. Admittedly, there are still many who remain sceptical about the potential of sophisticated pilotless aircraft. For them it should be instructive to look back to the recent past, as indeed it is for all who would be rash enough to attempt to predict the future, and consider how far we have come in the last 20 years or so. Retrospective contemplation, incidentally, reinforces the wisdom of Will Durant's dictum, "Those who do not know history are forever condemned to repeat it".

Little more than 20 years ago enemy territory would have been attacked by bombers at high altitude, carrying a crew of 2 or frequently more, with attrition rates that were regarded as acceptable. As far as computing was concerned, the first, primitive, pocket calculators were just appearing on the market and, by today's standards, they were horrendously expensive. In seeking to predict the progress of the next two decades it will be as well not to be too conservative.

Nowadays, as is widely appreciated, the military flying mission has become very much more demanding. Penetration at very low level is normal practice, the density of threats, both ground based and airborne, is exceedingly high and the amount of information concerning the mission, intelligence and threats that the pilot is expected to process is enormous. Continuing pressures to reduce crew numbers compound the problems by further increasing the demands on the remaining crew, often now just one man, and the shortage of time for reaching a decision and implementing it in low level flight makes matters still more difficult. Similar considerations apply to battlefield helicopters as to fixed wing fast-jet aircraft. Few would now dispute that the task of the human pilot in military aircraft has become so demanding that the provision of intelligent assistance will be essential in the near future. If we are to learn the lessons of history it must be seen as inevitable for the trend to continue and ultimately the pilot will be surplanted for some of those missions which today require manned aircraft.

In the information processing field the achievements in modern integrated circuits are too well known to require great elaboration. Such are the commercial pressures to achieve still greater component densities, storage capacity and processing speeds that there can be no real doubt that substantial further gains in all these areas will be made before the end of the century. In the immediate future current VLSI programmes will yield submicron integrated circuit technologies with useful increases in speed and component density. As long as silicon continues to be the favoured material VLSI developments will also continue to reduce the cost of any given data processing capability. Some discontinuity in this trend is probably inevitable as alternative, higher speed, materials such as GaAs reach the market place but in time they too will become cheaper. Whilst it is right to be optimistic, it is also necessary to appreciate that the pace of improvement may be slowing down as has been pointed out by Kuck et al [1]. Dramatic improvements in computing hardware will only come through significant changes in the basic technology employed but for this we may look to the advent of techniques such as parallel processing [1] [2] [3] [4]. Initially, useful gains will be made by configuring conventional systems in a parallel fashion but in the longer term the full potential of parallel processing will be realised through the use of technologies that lend themselves naturally to parallelism. Foremost amongst these at the present time are optical processing and optical data storage [5] [6] [7] [8]. However, it is not so much in the hardware that the real advances need to be made, although they should not be under-estimated, but in the associated software and here it is not merely the substantial programming problems that can currently be foreseen but the huge conceptual challenges that lie ahead that will prove the most awesome. Fortunately, we are armed with the new, scarcely tapped resource of Intelligent Knowledge Based Systems (IKBS) which is destined to be a formidable weapon in providing a worthy replacement for the man in the cockpit. Once again, although the challenge is daunting it would be fatal to over-emphasise the difficulties, we should take heart from the achievements of the past two decades.

At this point it is worth pausing to consider why the presence of a man in the cockpit is still readily accepted with little question. After all if an avionic engineer were to propose a sensor system of poor optical quality, connected by links of very low bandwidth to a processor with erratic performance that was never specified at all, let alone subjected to the highly formalised functional specifications that are regarded as a sine qua non in modern systems circles he would, I suggest, be unlikely to be taken very seriously. Yet that, with considerable simplification, is a fair description of the human visual system, admittedly redeemed somewhat by extremely sensitive photon detectors and by ill-understood but sophisticated focal 'plane' processing (for a more sophisticated account of human visual processing see, for example, ref 9). The basic data rates of human data processing and human reaction times are, by electronic standards, slow. So why is the man regarded as indispensible? Indeed why is it that man is, by common consent, still superior to any inanimate system? The reason is that, despite his short-comings, man remains the most effective parallel processor available. No image processing system can yet compete with the human operator neither can any man-made fusion technique yet approach the human ability to correlate information from widely disparate sources. In the field of signal acquisition and processing it is unlikely that any electronic system could currently match the extraordinary human selective capability manifested in the well known "cocktail party effect". Combined with these attributes are the well-established abilities of the human operator to draw upon a vast store of experience and to apply this in the form of heuristics to yield 'short cut', intuitive, solutions to problems. Man is also adaptable and versatile to a degree that no artificial system can yet match.

Granted these virtues, is it sensible to contemplate excluding the human pilot from the cockpit? Even accepting that he is currently subjected to extremely high workload would it not be better to confine our attention to providing him with a suite of intelli-gent aids that might enable him to undertake his task adequately? Even if this proved possible in the increasingly complex and hostile environment of future warfare there still remain the human physiological limitations that are likely to become an increasing drawback and there is the further point that eliminating the human pilot may also remove the necessity for many current constraints on aircraft design. The physiological factors include the well known high 'g' manoeuvre restrictions and the need to provide protection against the consequences of chemical and nuclear weapons. In this respect at least man is not an asset, he is a liability. Removing the man from the cockpit, of course, also removes the need for the cockpit itself and saves the bulk and weight of the ejection seat, displays and controls and eliminates the need for a large transparency. Deeper in the avionics significant savings would accrue from the elimination or reduction of redun-dancy introduced to ensure levels of safety that were only required for man-carrying aircraft. Environmental conditioning would no longer be required for the cabin but only for the avionics, with consequent reduced demands on engine air bleed and, finally, scarce front-end space should become available for important sensors. At the very least the savings in volume, weight, power dissipation and cooling requirements should be sufficient to offset the additional demands made by the new intelligent automated system.

2 WHY IS IKBS THE KEY?

Intelligent Knowledge Based Systems are the practical embodiment of artificial intelligence, which after years of being regarded as an academic curiosity, unlikely to be of practical importance, has emerged as almost a prerequisite for any system that purports to be up-to-date. What has happened to bring this about is a change in approach: instead of trying to create "intelligence" ab initio from basic components and rudimentary algorithms the philosophy is now to take existing human knowledge and embed this into specially designed computer environments. Instead of starting from scratch we are endeavouring to start from where we are now. This is the reason for the intense world-wide interest in so-called Expert Systems and their more sophisticated relations, IKBS.

At this point a word on nomenclature is appropriate; in the UK it is generally accepted that Expert Systems represent the simple end of the intelligent systems spectrum providing, for the most part deterministic answers, calling upon embedded human knowledge and often requiring a lot of user interaction. IKBS is the term reserved for more ambitious systems involving complex multi-reasoning systems and often involving much closer, direct interaction with the real world. This classification is illustrated in Figure 1.

FIGURE 1 INTELLIGENT SYSTEMS TERMINOLOGY

Increasing Sophistication ⟶

Expert Systems	IKBS	Artificial Intelligence
Often deterministic	Advanced, multiple reasoning	Intelligence
Require user interaction	Maintain multiple hypotheses	developed from
Generally small	Adaptive	fundamental
Simple reasoning	Near Real-time	abstract
Advisory	Large knowledge base	concepts
	Possibly self learning	Academic
	Possibly self extending	approaches

In principle, therefore, IKBS offer the opportunity to substitute an automated system for the human operator in the military aircraft cockpit. Clearly, this will initially be in the form of an intelligent assistant in a piloted aircraft [10] [11] [12] [13] but as confidence is gained and capability is extended the assistant will take over more and more so that eventually the pilot himself can be replaced. Whilst the IKBS techniques are still in their infancy they offer the valuable benefit of incorporating human knowledge and experience, in its most up-to-date form, into the automated systems. A powerful advantage of IKBS is their ability to maintain a consistent high level of performance without fear or fatigue and largely irrespective of the prevailing conditions. Furthermore, their embedded expertise may, if appropriate, be culled from more than one expert provided that the results can be suitably melded into a coherent body of knowledge. The problems of using multiple experts to generate the knowledge base are considered further later in the paper. In domains where expertise is scarce, possibly confined to a small number of people, IKBS offer the opportunity to capture the relevant knowledge, thereby ensuring that it is not lost and, if appropriate, that it can be more widely disseminated. Although this feature is generally thought of in the context of specialised terrestrial activities, it is in principle applicable to airborne skills (eg in ASW interpretation). It might also be valuable in training pilots to use new equipment, including of course IKBS themselves.

Granted that IKBS will make their way into the cockpit, initially in the role of intelligent assistants to the human pilot [10] [11] [12] [13], it is appropriate to consider what differentiates an intelligent aid from conventional automation. There are a number of key features, not all of which need necessarily feature in every system:

(i) Contextual Awareness

Conventional automated systems generally have closely defined limits of application. Indeed, formal functional definition procedures lay great store by precise, limiting demarcation of the domain in which the system is to operate and in defining in detail its interaction with other subsystems. By contrast IKBS boundaries may be fuzzy because the concepts they involve are not strictly defined. Their interactions may change dramatically with relatively slight changes in the circumstances. Very often an IKBS will be far broader in its scope than a deterministic system and it will be "aware" of far more of the context in which a decision is to be made and its decision will reflect that context, just as the decision of a human operator does. The decision not to attack a target of opportunity because it is of insufficient importance <u>bearing in mind the state of the battle</u>, would be just one such example.

(ii) Alternative Solutions

A well-established virtue of IKBS is their ability to present alternative solutions and recommended actions with confidence factors and, if desired, an account of the reasoning behind the course of action recommended. Alternative solutions may be presented in terms of various postulated developments in scenario, for example "If SAM site suspected at A is confirmed divert to ......" "If JPT continues to rise ......"

(iii) Self Learning/Self Extending

An IKBS should not be thought of as immutable, after all a human pilot continually learns by experience. Thus a sophisticated IKBS will involve feedback from the world with which it is interacting and will have the ability to change its reasoning accordingly. In the broader context of mission planning the IKBS may be expected to report on its experience during the mission, including such information as a high level interpretation of threat data. Suitably combined with that from other aircraft this would modify the database of the IKBS for subsequent missions.

(iv) Adaptability

In many cases IKBS incorporate a user model and there is no reason why this should take a single form. It should certainly reflect the experience, ability and characteristics of the pilot so that, for example, it would not trouble a highly experienced pilot with unnecessary detail but would provide this for one less experienced. It might also compensate for individual characteristics, a particular pilot's over-enthusiasm for active jamming might be an example. Personal characteristics, including preferences, could easily be fed into the IKBS with pre-mission briefing material (an extremely simple precursor to this is the driver preference data on seat and mirror positions now stored in more expensive cars). Again the IKBS should adapt to the increasing experience of the pilot, even in the course of the mission and once more the analogy with the behaviour of a human companion is relevant.

(v) Explanation

One characteristic of expert systems that is widely acclaimed is the ability to provide explanations in varying degrees of detail of the reasoning underlying their decisions or recommendations. Whilst this may well be of value in non-real-time terrestrial applications, particularly those in which there is a high degree of user interaction (the many well-known medical diagnosis programs are typical examples), it is a technique that is likely to be of only limited value in military aircraft. Certainly, there is rarely time for the pilot to recognise that he has a need for an explanation and for him to request it and digest it before action needs to be taken. Thus the system must be designed to provide the right degree of explanation and justification, but no more, and to present it in a readily assimilable form. Furthermore, it must adapt its explanation to the circumstances, curtailing it when time is too short or when too much other information must be conveyed simultaneously. A truly adaptive IKBS will adjust the extent and depth of explanation to suit the experience and requirements of the pilot and, moreover, will continuously readjust them as the mission progresses.

It is evident that an IKBS should be far more adaptable and compliant than normal software systems. As progress is made towards fully autonomous unmanned aircraft it is anticipated that the IKBS provided as intelligent assistants will evolve toward a more autonomous role. Much of this evolution may occur in a straightforward way, without the need to completely re-think the systems concepts. Faith in the potential of fully autonomous systems will be generated as confidence in IKBS techniques is built up during the phase in which they fulfil the role of intelligent assistants. It follows that some care must be taken over the introduction of the initial, discrete IKBS in an advisory, aiding capacity, since it would be quite easy to undermine confidence, rather than create it, either through the use of systems that purported to be intelligent but which turned out to be dim-witted or, conversely, by premature attempts at over-ambitious systems which failed to live up to their aims. Once again the lessons of the past, not least those learnt from the introduction of computing equipment into military aircraft, must be firmly in mind.

3 POTENTIAL APPLICATIONS OF IKBS

Having agreed that progress towards the unmanned cockpit will be evolutionary and that it will follow on from the development of discrete intelligent aids of gradually increasing scope, it is appropriate to consider what applications in the military cockpit are most likely to benefit from the use of IKBS. It is convenient to consider the potential applications of IKBS in military aircraft under a number of headings:

3.1 Planning

Planning is a major activity both prior to a mission and whilst it is in progress. It is a splendid example of an intelligent reasoning process that calls upon many sources of data, often otherwise unrelated, and which almost invariably requires knowledge of the context. The time constraints for pre-mission planning are, of course, generally somewhat less severe than those for planning in the course of the mission, otherwise the two aspects are broadly similar.

Mission planning must take account of a wide range of factors. Whilst the basic objective is often well defined and of the "Destroy fixed target at point X" form, even such a definitive objective may be subject to caveats (eg "at all costs" or "without placing the aircraft at risk" or "assuming that it has not been successfully attacked by other aircraft beforehand"). The objective will often be subject to change in the course of the mission and for many missions, CAP or attack of battlefield armour by either helicopters or fixed wing aircraft, for example, it will not be precisely defined at the outset. In these cases there are many implied constraints on the mission that the pilot has knowledge of, possibly subconsciously, that must be emulated by an IKBS if it is to approach similar levels of performance. (Of course, it may be that not all the pilots preconceptions are beneficial ......) Once the basic objectives of the mission and the related constraints are decided a choice of route must be made and this aspect of mission planning has already received considerable attention from the IKBS community [14] [15]. Numerous factors impinge upon the planning of a suitable route. Account must be taken of the terrain and of known or suspected enemy threats. If the data is uncertain the reasoning process must assign a probability to its validity and take account of this in optimising the route. Since it is only to be expected that the enemy will dispose his defensive missiles and armament in greatest concentration along just those routes that seem most attractive topographically, the choice of optimum route is unlikely to be self-evident. It may also be influenced by the presence of other friendly aircraft in the area concerned. Mission planning must also take account of such factors as weather, the availability of diversionary airfields, aircraft parameters, the likely role of enemy fighters, chemical contamination and any other aircraft taking part in the attack.

Mission planning is known to be a time consuming process, even when conducted before the mission starts. For battlefield helicopters it is frequently necessary to re-plan at short notice in the course of a mission in order to take account of changes in the state of the battle or to cope with urgent or emergent threats. In this case all the above factors must still be taken into consideration but the decision must be arrived at with great rapidity. The choice between a direct route through densely defended terrain and a lengthier, less threatened route which takes longer and requires more fuel, may not be easy to gauge correctly in an instant in the heat of battle, especially when the data is incomplete, out of date, uncertain or conflicting. In such applications IKBS already clearly have an important role to play, especially since the aircraft now being developed will have fewer crew than their predecessors. It may also be foreseen that from these relatively straightforward route planning aids, which take account of only a limited number of the factors considered above, will evolve more complete mission management systems capable of assuming much of the mission planning responsibility. A major factor to be considered in their development is the manner in which intelligence data, on enemy threat locations, for example, is updated, since the manual insertion of large quantities of data by the pilot in the course of the mission is unlikely to be acceptable. This question is not straightforward since it involves more than just the mechanics of data entry - itself no trivial matter - but extends to consideration of the extent to which the pilot believes the information - in the most obvious case, because of his position he may know that a particular piece of data is out of date, but more subtle examples may also arise. It would be counter-productive to insist on the pilot vetting each and every incoming piece of information. Many questions of this sort will arise as IKBS planning aids evolve and will only be answered in the light of experience.

Planning is a topic of major interest to the IKBS community at large and it is likely that military airborne applications will benefit extensively from progress in the civil field, from such applications as the development of automated factories, autonomous land vehicles and others. Few civil applications, however, are as demanding as their military counterparts, especially as regards the need for near real-time operation.

Whilst the present discussion has centred on mission planning in aircraft, since this is a clear element in the march towards the unmanned cockpit, it should not be forgotten that other aspects of air-warfare depend upon planning activity, often at high level. Obvious examples are the disposition of aircraft, either for offensive or defensive operations and, on the battlefield, the tactical disposition of helicopters. It is to be hoped that much commonality will exist between ideas developed to solve planning problems in these areas and those in the airborne environment.

3.2 Signal Processing and Data Fusion

One of the striking features of the development of military aircraft since the Second World War has been the dramatic increase in the number and variety of sensor systems that have become standard equipment, although, in passing, it is interesting to note that primitive radar warning receivers were carried by many aircraft in the later stages of WWII. Most parts of the electromagnetic spectrum can now be detected by military aircraft, with sensors covering radio, radar, infrared and visible frequencies, although in the latter case the sensor is usually the human eye. Added to the data provided by the aircraft's sensors is a mass of information relayed from other aircraft and from the ground. One of the principle functions of the human pilot, and for which he is at present uniquely capable, is the integration of all this data. He must, for example, decipher the indications from his threat warning sensors, generally at a time when they are indicating a large number of possible threats, and correlate the information with intelligence data received on the ground prior to the flight and, subsequently, in the course of the mission. In some cases this may need to be further correlated with topographical data and with radar returns in order to build up a complete picture. On occasion the decision to use an active radar sensor, for example, may depend on the initial sensor data correlation.

Underlying this massive task of data fusion is the fundamental and related issue of signal processing. Where the information is pictorial, in seeking target information in the output of a FLIR, for example, the human visual system still reigns supreme despite, as noted earlier, its optical imperfections and its inherently low bandwidth. The human ability to perform visual target detection and recognition better than a machine almost certainly owes much to the application of heuristics derived from experience and this is therefore an obvious and natural application for IKBS. However, since many of the heuristics involved are probably not formulated consciously but are held at a deeper level in the brain, the problem of knowledge elicitation is likely to be unusually formidable. Thus, in signal processing, there is currently a fairly clear division between information conveyed by the normal human sensory modalities, primarily vision and hearing, in which the human signal processing is clearly superior and which IKBS must strive, with difficulty, to emulate, and those sensors which detect electromagnetic signals far removed from normal human ranges and present their outputs in formats which, although visual or aural, are not a familiar part of normal human experience. There is no way in which a human being would consider de-interleaving the mass of complex signals handled by an ESM receiver, even if he possessed the necessary sensors. Thus for such data the electronic system is already accepted as indispensable [16]. It is worth noting, however, that the interface between the electronic system and the human operator is still far from satisfactory. Despite the inherently good performance of the human visual system in processing pictorial information we have yet to devise an interface with an ESM system, which although complex produces far less information than a normal visual field, which handles the latter data as effectively. Perhaps this problem will only disappear with the advent of far more intelligent processing.

Whilst signal processing has been a large and active research area since long before the advent of IKBS, stimulated by such applications as automatic image processing for robotics among many others, data fusion as an issue has come to the fore with the development of IKBS - possibly because it is, intrinsically, an intelligent activity. Civilian applications do exist, in such vital areas as the control system for complex industrial installations such as power stations and also in process control, but once again the extreme demands of military systems, especially in the real-time environment of the military cockpit are likely to provide the greatest challenge.

Indeed a major step forward in the development of IKBS architectures, the development of the "blackboard" configuration arose through the HASP/SIAP programme in the USA which was applied to the interpretation of sonar data [17] and this remains amongst the best known of IKBS signal processing/data fusion applications. (Strictly speaking the blackboard architecture was not invented for HASP/SIAP, it had evolved earlier for the HEARSAY speech recognition/synthesis programme, nonetheless the challenge of the military signal processing/data fusion application provided the simulus for further development [18].)

3.3 Intelligent System Monitoring

Fundamentally, system monitoring requires little intelligence. Despite this, the crew of present aircraft are expected to spend much of their time maintaining a watch on the aircraft systems and dealing with indicated malfunctions, again often in a well prescribed routine fashion. In the event of there being too many prescriptions to remember, they may be provided with flip-cards for reference. System monitoring requires intelligence when the appropriate action to be taken is dependent upon the circumstances or where the sensor data is ambiguous or imprecise and heuristic knowledge is called upon in the interpretation. A particular equipment may be known to exhibit certain fault characteristics in given circumstances or diversionary landing sites might entail disadvantages (lack of facilities, proximity to enemy forces, etc). Thus, whilst some faults are simple, say a generator failure, others may give rise to a multitude of complex indications that are difficult to decipher. A serious engine failure, due to malfunction or to enemy action, might well trigger warning indications relating to hydraulics, fuel and electrical systems. The rapid assessment of such a plethora of warnings calls for an *intelligent system*. Certainly the use of reference material, be it flip-cards or an interactive "Expert System" is impractical at a time when the pilot needs to take urgent action and needs to devote all his effort to controlling the aircraft. This illustration also provides an example of the value of contextual awareness; the possibility of an otherwise improbable malfunction due to damage from enemy weapons would clearly be afforded a much higher probability when the aircraft was flying over enemy-held territory.

However, since failure analysis is often either deterministic or involves fairly straightforward probability reasoning, the system may often tend to divide naturally into two sections, the first performing the analysis in a fairly well-defined deterministic way whilst the second (embodying the intelligence) decides upon the actions to recommend to the pilot.

Failure analysis systems are important to the development of IKBS and therefore to its evolution towards the goal of achieving autonomous unmanned operation, since they include many of the straightforward and, more important, the largely self-contained systems. Simple failure analysis systems with a modicum of intelligence are practicable now and through consistent accurate diagnosis and recommendation should help significantly to lay the foundation for the confidence in IKBS that will be crucial to their acceptance.

Almost any failure analysis system will be called upon to process data with a wide range of priorities and leading to conclusions and required actions of varying urgency. It must be able to recognise the important conclusions, especially when, as is often the case, several system failures are being dealt with simultaneously. Morever, it must present its conclusions and actions in order of priority. The ability of IKBS to maintain a number of alternative hypotheses throughout the reasoning process is one feature that makes them a natural choice for failure analysis applications. This is also of considerable value in recognising that two or more failures being treated as independent may, in fact, have a common origin. IKBS continuously accepting updated information from the sensors in near real-time should show considerable advantages over basic "snap shot" indicators.

In the longer term more sophisticated IKBS failure analysis systems will not only assess the priorities of the failures with which they are dealing but will have sufficient awareness of the situation of the aircraft and its mission, possibly because they form part of a much more extensive mission management system, to calibrate the importance of the failure information in terms of other mission priorities. Some failure indications that would otherwise be presented might be withheld during critical parts of the attack phase, for example. Of course, eventually the IKBS will be able to decide whether it should initiate action itself or inform the pilot.

System monitoring does not simply mean failure analysis and there are many chores (eg fuel management) that automatic systems will take over from the pilot. Indeed this is already happening. In the field of flight control IKBS have the potential to extend current manoeuvre demand techniques to include consideration of the context in which the manoeuvre is to take place. The rapid consumption of fatique life becomes of little consequence if the alternative is to lose the aircraft either as the result of attack or system failure and a human pilot would not hesitate in such decision. An IKBS should do likewise. It should also be good at choosing the optimum way of achieving a given manoeuvre, again through awareness of the context.

3.4 Intelligent Aids

There are a number of simple discrete aids, incorporating a modest degree of intelligence, that are likely to be among the earliest examples of IKBS to appear in the cockpit and which can conveniently be considered as a group. The application to automatic display formatting and display surface allocation is one such example. This may range from fairly straightforward, stereotyped display configuration to more ambitious arrangements which take account of the likelihood of the information being needed at any given point in the mission and of the extent and format in which it should be provided. Thus the scaling of a map display may be simply tailored to suit the anticipated requirements of the aircrew, it being unlikely that a pilot flying at high altitude would be able to make use of very detailed topographical information, for example. Essentially, this amounts to automatic, intelligent decluttering; clearly it must be intelligent since pilots will have little sympathy with a system that removes information just as they are attempting to use it. Neither will there be much support for a system that is subject to frequent changes in pursuit of an "optimum", and clearly much care will be needed in the design of such systems. One feature which may be desirable is an indication, following a change, of where the information has gone or how it may be retrieved. Initially, automatic display configuration will be applied to the reversion necessary in the case of failure in the display system or in the case of an emergency in the aircraft. Control of display formats through direct vocal input will probably be one of the earliest applications of speech recognition technology in the cockpit.

Under the general heading of resource allocation a number of promising applications for IKBS may emerge, including intelligent aids in several diverse fields. Communications management is becoming more onerous with ever increasing emphasis upon covert operation and a system that makes the best use of communication channels in terms of propagation constraints, intelligibility, etc could clearly embody a certain amount of intelligence and relieve the pilot of an appreciable element of his workload.

Resource allocation in the anti-submarine warfare (ASW) field includes such considerations as the decision to deploy sonobuoys in given configurations and types in terms of the prevailing conditions, including the needs of the mission and, in the case of an ASW helicopter such parameters as its fuel remaining. The extent of the intelligence embodied in such systems depends largely upon the degree of awareness of the situation built into the IKBS.

Considerations of resource allocation also enter into the ECM and ESM fields when the decision to use jammers or chaff arise and, in the broader context, in the allocation of ESM resources. *Again, contextual awareness is important and role of IKBS will depend* upon the extent to which this is incorporated or to which the specialised aids are able to interact with more extensive systems with greater awareness.

There are clearly a very large number of potential applications for IKBS in the cockpits of fixed wing fast-jet aircraft and in helicopters in all their diverse roles. Most of these will be undertaken individually as IKBS becomes an ever more practical reality and they will provide invaluable experience upon which the longer term programme to develop fully autonomous unmanned aircraft can capitalise. Further experience will come from the numerous IKBS being developed worldwide for commercial and civil use and from other military applications, most directly from other autonomous vehicle programmes (the DARPA autonomous unmanned land vehicle programme being a notable example).

4 THE PRACTICAL APPLICATION OF IKBS

4.1 Present State of IKBS

This is not the place to embark upon a review of achievements in IKBS, nor is it necessary since there are a number of accounts that are generally available [19]. Already there are substantial achievements in the field but, as perhaps is to be expected, they are mostly at the "Expert Systems" end of the spectrum. It is probably still true to say that there has yet to be seen a system that even a majority of observers would agree to call "intelligent". Most do not purport to be so, their proponents would merely claim that they are useful, often as good as their human counterparts, occasionally better, and that they achieve that performance by the incorporation of human expertise. It is not unusual for an expert system to yield a deterministic answer but, having employed heuristic techniques, to produce it far quicker than a conventional program. Perhaps it is worth remarking, however, on the paradox that becomes apparent when an intelligent system is described. Invariably, considerable effort is devoted to producing an account that makes clear every essential detail of the system and once it is so **described it seems**, not unnaturally, virtually self evident and therefore, not intelligent. There is some truth in the suggestion that something which can be described completely cannot be intelligent. For most IKBS worthy of the name, of course, complete description is in reality not practicable, when details of the reasoning processes are borne in mind, however, descriptions given in terms of their inputs and outputs with a basic outline of the reasoning process ("it involves a rule base of some 500 rules") sound as if they are complete and perhaps this is where the difficulty often lies.

Whilst this may seem a digression it must be remembered that the goal of introducing intelligent aids into the cockpit will only be achieved with the willing co-operation of the pilots who will use them, and it may be assumed that they will already have encountered demonstrations of "intelligence" that left them unconvinced.

Although there are a number of operational expert systems in the commercial world, they make poor examples of intelligence. Worse still, most of them are interactive and require extensive user input. This is completely unacceptable for airborne applications in all but the very simplest cases. More sophisticated systems, true IKBS, do exist but they are almost all at the research stage. It is interesting to note that many of the more advanced IKBS are aimed at military applications. There have as yet been no reports of IKBS operating in an airborne environment.

Most simple systems employ relatively conventional backward chaining logic and it is almost a hallmark of a true IKBS that its reasoning system is far more flexible. At the very least it will incorporate both forward and backward chaining but the most probable arrangement is the so-called "Blackboard" [18] referred to in section 3.2. In its simplest form this consists of a central element (the blackboard) with connections to surrounding surrogate entities, knowledge sources relating to particular aspects of the problem, 'demons' for tackling certain functions, knowledge bases, etc. Basically the idea is that the current state of the problem is maintained on the blackboard and the peripherals feed their outputs to this when called upon or when they have some useful contribution to make to the solution of the problem. Conversely, they can indicate needs that they may have for particular pieces of data before further progress can be made. Various blackboard configurations have been proposed, including multiple blackboard systems, but they share the common virtues of great flexibility, incremental problem solution in an opportunistic (ie not previously defined) fashion, and the natural incorporation of data flows from the real world. In principle, they should be fairly readily extensible. As with most IKBS they tend to be extravagent in their demands for computer processing capacity and for data storage.

4.2 Developing Practical IKBS for Military Aircraft

From the earlier discussion in this paper it will be apparent that two features distinguish military IKBS from ordinary commercial applications of the technology. These two factors are, firstly, the need for near real-time operation and, secondly, considerabl complexity. Each of these represent a considerable challenge to the researcher for his solution must not only satisfy the functional need it must also be sufficiently practical for use in military aircraft. Thus, the implementation must be on hardware that is rugged, compact, reliable and affordable besides having sufficient speed, power and storage to meet the need. Neither should it require excessive amounts of electrical power or place undue strain on the environmental cooling systems. Input and output requirements must be compatible with the aircraft systems and software interfaces must be designed to be fully compatible with the aircraft's conventional data and computing systems. To a considerable extent these practical factors increase the difficulty of achieving near real-time operation in complex systems.

Even in conventional systems the requirement for near real-time operation represents a challenge; in IKBS the difficulties are multiplied. It is not merely a matter of ensuring that data inputs and outputs are sufficiently rapid and well organised and that the processing is fast enough. In IKBS there is the significant additional and fundamental problem that continuous reasoning over time varying situations is a topic which has yet to receive much attention in the IKBS world. Recalling that IKBS may be self-extending and highly adaptive, places this difficulty in perspective. Although there are a number of high-level toolkits that are powerful aids to the prototyping of IKBS, such as ART, KEE, LOOPS, PICON and others, none of these was designed for the complex, real-time systems needed for military airborne applications. Thus the Royal Aircraft Establishment (RAE) at Farnborough in the UK has funded Cambridge Consultants Limited to develop a toolkit specifically designed to meet such needs, it is called MUSE.

4.3 MUSE - A Toolkit for Real-Time IKBS

MUSE is designed to support the prototype development of military airborne IKBS applications and it has two essential ingredients, a powerful set of software tools for prototype system development and a compact, solid state target machine capable of being flown in aircraft for system evaluation. The choice of hardware for the experimental system and for the prototyping machines was strongly influenced by the need to facilitate its use by a large number of teams tackling a wide range of applications. Thus the development system is based upon the SUN workstation upon which the basic prototyping is undertaken with the compiled code being subsequently downloaded to the target machine which employs a 68000 series processor.

MUSE is designed to be flexible and it includes a range of representation languages that may be used in combination for maximum efficiency. The flexibility extends to the reasoning system which involves separate reasoning modules, communicating by shared access to particular databases. The number of and relations between these modules and databases is not fixed. It is anticipated that the system will mostly be used in a generalised blackboard format (section 4.1) and the approach employes the 'object oriented' concepts that are well developed. Each knowledge source is an 'object' in the system parlance (as are a number of other quantities), and contains a local rule structure (with its own private database to act as fast working memory) in addition to its own database, which is accessible to outside objects. This seemingly complex structure allows for considerable flexibility in system construction and operation whilst maintaining high speed. A more detailed description has been given by Reynolds [20].

In the MUSE hardware moving parts have been excluded in the interest of ruggedness and reliability. Non-volatile memory is used to hold the IKBS code and it also records events during trials for subsequent analysis. Facilities are provided for accepting aircraft and sensor data and for suitable outputs to displays and aircraft systems. The entire system, which employs only standard commercial components, fits into a single box of modest proportions.

MUSE has already been applied in the laboratory to a simple cockpit warning system and it is hoped to have it flying in an RAE helicopter in late 1987.

4.4 Future Role of Parallelism

The scale of IKBS considered in this paper with its ultimate goal of replacing the man in the military aircraft cockpit is far greater than any existing IKBS and it is already clear that its implementation will make enormous demands upon computing technology both as regards processing power and data storage. As already remarked this implies the need for parallel processing and for optical data storage and processing. Crude parallel processing is already with us, some data handling problems being readily amenable to straightforward parallel techniques (problems which involve many repetitions of the same calculation and which benefit from SIMD architectures are the simplest examples; somewhat more complex are MIMD configurations, which require the problem to be conveniently divisible amongst processors). IKBS reasoning programmes, however, follow ill-defined paths that may change dramatically in form and extent as a result of quite small variations in the input parameters. They are thus not readily amenable to predetermined subdivision in order to allocate portions of the problem to different processors. With IKBS the efficient use of parallel processors will necessitate a wholly new outlook and this is bound to include the adaptive use of processing power. Whilst this is quite straightforward on a modest scale, when considered on a large scale (say 1000 processors) it represents a conceptual challenge of mind-boggling proportions. It may well be that, initially, parallel processing will be applied to blackboard-like architectures, perhaps with one processor to each knowledge source or, more economically, with a number of processors ready to undertake processing for individual knowledge sources as their needs arise.

A somewhat separate application of parallel processing is in the domain of self learning, self adjusting configurations such as neural networks and work is under way in this field [21]. Although this is still at an early stage it is a powerful concept that is bound to find widespread application in the IKBS field.

4.5 Validation

Those concerned with aircraft programmes are already acutely conscious of the problem of validation. Quite severe problems are already encountered in the validation of large suites of conventional software and it is therefore only to be expected that when it comes to IKBS, reasoning with uncertain data and possibly including self learning or self extending capabilities, the problems will become still more difficult. Many lessons have been learnt in the development of conventional programming and obviously these should be applied as far as possible to IKBS. It should for example, often be possible to attempt functional definitions at the outset of development, although this is more feasible for the simpler, more nearly deterministic systems. With more advanced IKBS, whose very nature and the basis for whose importance is their generality and adaptability, only the broadest of definitions will be possible when the work is undertaken [22].

As far as possible internal consistency must be maintained in the course of system development and there are well established procedures for the control of conventional software development which can be adapted to apply to some limited extent to IKBS programs. However, once again the very characteristics of IKBS which make them attractive for the solution of problems requiring intelligent reasoning, namely their vast choice of possible solution paths and their ability to adapt their own reasoning and modify both their reasoning processes and their databases, are by their very nature antipathetic to validation. Thus, whilst it is obviously sensible to take all reasonable steps to validate an IKBS, it must be recognised that comprehensive validation will be unattainable either through the application of formal procedures or by the application of test cases since it is unlikely to be practicable to administer a sufficiently wide range of appropriate trials.

Against this gloomy background it is important to retain a sense of proportion. It is all too easy to call for standards of reliability, accuracy and validity from a computer system that are far higher than those demanded of a human operator. Yet the

IKBS may be doing no more than taking over part of the function of the human pilot. If, for the moment we regard the military pilot as a "system" it is worth pausing to consider the extent to which he has been "validated" at the completion of his training. For such an ill-defined system the validation may be considered rather limited. Apart from the theoretical training it consists of the administration of a relatively small number of "test cases" none of which apply to the situation in which the peak performance is required (ie in the stress and confusion of war rather than peace-time training) and some of which are conducted not in flight but in a simulator. It could never be claimed that the envelope of human piloting activity had been more than sampled to a limited extent. Furthermore the measures of performance applied are relatively few and to a large extent subjective. To cap it all it may be remarked that the assessment 'systems' are themselves neither calibrated nor validated. In defence of the human operator it may reasonably be countered that, for all that, the 'system' appears to work and achieve its objectives quite well. The credit for this must be attributed to the high degree of adaptability exhibited by human beings, in other words man is able to compensate for his own shortcomings. Such thinking may provide a valuable clue to the way in which progress towards the validation of IKBS may be made.

Thus, the first important step is the realisation and acceptance that the anything that would remotely approach an acceptable standard of formal validation, in the conventionally understood sense, is, for IKBS, impossible. Secondly, it should be recognised that if current standards of human performance are regarded as acceptable, there is no reason why higher standards should be required of intelligent systems (although, of course, they are highly desirable), provided that the IKBS achieve those similar levels of performance overall and not merely in some artificially limited domain. Since the human pilot achieves his acceptable overall performance through an ability to adapt and to cope with his own shortcomings why should not IKBS, which are also highly adaptive, emulate the self-assessment and self correction of the human being? Hence conventional formal validation would be replaced or at least supplemented by self-validation. In order to achieve this IKBS will need to apply criteria of "reasonableness" to evolving solutions, indeed it is possible to visualise the development of concepts that are not domain-specific but which can be used to provide an indication that solutions are developing within appropriate and acceptable bounds. Such a module, once developed, might be quite widely applicable to a broad range of intelligent systems - conventional numerical tests could be included where appropriate and would form a natural subset of the symbolic processing procedures. By this means IKBS could, in principle, be validated to give performance of at least similar standard to that attained by human operators.

4.6 Knowledge Acquisition.

It is already a matter of common experience in the development of ordinary expert systems that knowledge acquisition can represent a major problem. Indeed, validation, in the sense of confirmation that the knowledge actually built into the system truly represents that culled from the human expert, is also a problem in knowledge acquisition. However, for the present, it is reasonable to trust that that will be solved as far as is possible by careful procedural checks. The less tractable issues arise from the widely different approaches adopted by equally competent experts, from the need to ensure that all the relevant knowledge of the experts is explored [23] and in relating the knowledge to the level of expertise that the IKBS assumes in the pilots with whom it is to interact.

Incompatibility between experts is a widely recognised problem in IKBS and airborne applications are in no way exempt, as is implicit in the time-honoured dictum, "If you want 3 opinions try asking 2 pilots". At first the temptation is to attempt to "average" in some fashion, the different approaches. Assuming that the approaches are genuinely irreconcilable, this procedure is, by definition, doomed to failure. Furthermore, the idea of applying 'cut and splice' techniques, adopting one expert's approach for part of the problem and later changing to another's, clearly holds dangers of discontinuities, possibly quite subtle in nature, that should spell caution in other than quite special circumstances. Thus, once irreconcilable differences have been established a decision to adopt the approach of a single expert rather than an amalgam of several is likely to be preferable. This could be viewed as an elitist rather than a committee-based approach. It does, of course, imply the need for exceptional care in the choice of expert.

Often, however, the choice may lie between expert approaches that are of equal standing but which differ strongly in character. One could think of flying styles that might be summed up as, say, 'vigorous' or 'methodical', for example, equivalent but very different. Clearly, just as one style is the hallmark of a particular expert so it will suit pilots who feel an affinity for that type of approach. Conversely, the other extreme they might find irritating. Yet by relying upon one or other expert in building the knowledge base such characteristics would inevitably become built into the system - eventually a sophisticated IKBS might be said to incorporate some elements of a personality. It would be counterproductive to inflict upon a pilot an IKBS with a 'personality' that was alien to him and the remedy might be to provide alternatives - either contained within a single IKBS or as options selected when the system is briefed before the start of a mission.

If an IKBS is to be accepted as a genuinely intelligent aid it must command the respect normally accorded to an intelligent companion. In particular it should appreciate any change in the experience or expertise of the pilot and modify its interaction accordingly. To enable it to do this means must be provided for it to monitor the pilot's characteristics in some way. Whilst this represents something of a challenge it is likely to be essential if IKBS are to be regarded as anything but rather dim-witted, after all no-one would have much time for a human companion who took no note of increases in experience, or perhaps, of the fact that a similar event has already occurred earlier in the mission.

Ensuring that all the relevant knowledge and especially that all appropriate heuristics are culled from the expert and incorporated in the IKBS, is a seemingly obvious consideration and yet it is a potentially serious error that could be both common and unappreciated. Quite often an expert is unaware of the heuristics he employs and indeed it is not unusual for an expert to have an incomplete or erroneous idea of the way he tackles a task. Such factors are well known pitfalls in knowledge acquisition. One virtue of using more than one expert, at least initially, is that it may reveal incompleteness in the knowledge base derived from other experts.

In a large and complex domain the use of several different experts to cover individual aspects of the problem is inevitable and this will clearly be the case for any ambitious intelligent cockpit system, with separate experts covering control, ESM, navigation, weapons, etc. Not only does this require especial care when defining interfaces it also entails the need for special vigilance to ensure that nothing at the edges of a domain is inadvertently omitted.

5 SUMMARY - THE ROAD TOWARDS UNMANNED AIRCRAFT

In the course of this paper we have examined the thinking that leads towards unmanned aircraft and by looking at the breathtaking pace of progress in IKBS, computing and electronic data handling have confirmed that the technology required to achieve intelligent autonomous unmanned aircraft is likely to be achievable.

Fortunately, an evolutionary course is both possible and desirable. Indeed, progress towards unmanned aircraft involves discrete steps that are of such modest proportions as to form a quasi-continuum. No great breakthroughs are called for but, nonetheless, the sum total of the discrete steps constitutes a formidable challenge. Thus the path lies through initially discrete, relatively simple systems with relatively modest pretensions to intelligence but which build confidence and provide experience in IKBS, to more ambitious systems which interact strongly amongst themselves and with the traditional deterministic aircraft systems to provide the more complete contextual awareness that is one of the hallmarks of a sophisticated intelligent system. As further confidence is gained greater autonomy will be granted to the IKBS and fewer decisions will be referred to the pilot. The way to complete autonomy and unmanned aircraft is then clear.

REFERENCES

The IKBS literature is already vast. In most cases, therefore, the references given are typical examples of sources:

1 KUCK, D J, DAVIDSON, E S, LAWRIE, D H and SAMEH, A H, "Parallel Supercomputing Today and the CEDAR approach" SCIENCE, 231, 961 (1986)

2 GUPTA, A, "Parallelism in production systems - the sources and the expected speed-up" in Expert Systems and their applications, 5th International Workshop, 1, 25 (1985)

3 HALSTEAD, R H, "Parallel Symbolic Computing" COMPUTER, August 1986, p35

4 GABRIEL, R P, "Massively Parallel Computers: The Connection Machine and Non-Von" SCIENCE, 231, 975 (1986)

5 See, for example, special issues of APPLIED OPTICS such as; 25 No 10 (May 1986); 25 No 14 (July 1986); 25 No 18 (September 1986)

6 EICHMAN, G E and CAULFIELD, H J, "Optical learning (inference) machines" APPLIED OPTICS 24, 2051 (1985)

7 SMITH, S D, "Lasers, non-linear optics and optical computers" NATURE 316, 319 (1985)

8 MADA, H, "Architecture for optical computing using holographic associative memories" APPLIED OPTICS 24, 2063 (1985)

9 BURTON, G J, HAIG, N D and MOORHEAD, I R, "A self-similar stack model for human and machine vision" BIOL CYBERN 53, 397

10 POHLMAN, L D and PAYNE, J R, "Pilots Associate Demonstration One: A look back and ahead" IEEE, NAECON Proceedings 1986, p1186

11 SHELNUTT, J B, STENERSON, R O, NELSON, P C and MARKS, P S, "Pilots Associate Demonstration One: A look inside" IEEE, NAECON Proceedings 1986, p1184

12 KANDEBO, S W, "Lockheed Stresses Crew Participation in Pilots Associate Development" AVIATION WEEK and SPACE TECHNOLOGY 7 July 1986, p111

13 MORISHIGE, R I and RETELLE, J, "Air Combat and Artificial Intelligence, AIR FORCE MAGAZINE, October 1985, p91

14 GILMORE, J F, SEMECO, A C and EAMSHERANGKOON, P, "Knowledge-based route planning through natural terrain" SPIE 548, 128 (1985)

15 FRANKOVITCH, K, PEDERSON, K and BERNSTEEN, S, "Expert Systems Applications to the Cockpit of the 90s" IEEE AES Magazine January 1986, p13

16 WILBY, A and RICHARDSON, D, "Advanced Technologies Play Key Role in Designing Future EW Systems" MSN and CT *September* 1985, p66

17 NII, H P, FEIGENBAUM,E A, ANTON, J J and ROCKMORE, A J, "Signal to Symbol Transformation: HASP/SIAP Case Study" *AI Magazine Spring* 1982, p23

18 NII, H P, "The Blackboard Model of Problem Solving" AI Magazine, Summer 1986 p38

19 See, for example, BARR, A and FEIGENBAUM,E A, "The Handbook of Artificial Intelligence" Vols I, II, III W KAUFMAN, INC (1981)

20 REYNOLDS, D, "From Development to Delivery - System Integration" KBS-86 (Online Publications) p301, (1986)

21 TAKEDA, M and GOODMAN, J W, "Neural networks for computation: number representations and programming complexity" APPLIED OPTICS 25, 3033 (1986)

22 SMITH, D E and SEEMAN, S E, "Application of Automated Reasoning Software: Procedure Generation System Verifier" ANS/ENS Fast Reactor Safety Meeting Knoxville, USA April 1985 p161

23 GAGLIO, S, MINCIARDI, R and PULIAFITO, P P, "Multiperson Decision Aspects in the Construction of Expert Systems" IEEE Trans SMC-15 536 (1985)

GENERATING DIAGNOSTIC KNOWLEDGE FROM A STRUCTURAL AND FUNCTIONAL DESCRIPTION *

Jean-Pascal AUBERT
Jean-Michel CORNILLE
Alain MELLER

Laboratoires de Marcoussis
Centre de Recherches de la C.G.E.
Route de Nozay
91460 - MARCOUSSIS
FRANCE

ABSTRACT

Our paper deals with the problem of diagnostic knowledge acquisition. We present a front-end to a diagnostic system under development, that provides aides to the expert in the process of diagnosis rules design. This front-end uses *structural and functional* modeling. It seems to be general enough to be applied to other applications that we characterize.

1. Introduction

Our application concerns the maintenance of an aircraft Navigation System (NS). This system is divided into physical units, called LRUs (Line Replaceable Unit). When the pilot has noticed something wrong during a flight, the ground technician must find and change the failed LRUs to get the aircraft back in an operational state. Maintenance is quick if he is able to find failures by making simple tests by himself. The aim of the expert system is to assist the technician in his job in order to avoid the use of a specialized processor which tests the plane systematically but immobilises the plane for a long period.

1.1. Previous work

A first version of the expert system was created [2]. Its knowledge-base was made up of :

(1) knowledge about the LRUs, such as the replacement cost and the probabilities of known failures,
(2) knowledge about the tests including the cost of operating and diagnosis rules in the form :

(f0) conjunction of tests results ⟹ conclusion

The conclusion part either suspects or discards possible component failures.

The general strategy used consists in choosing the operation (test or replacement) that minimizes the average probabilistic cost of a repairing session.

1.2. Criticisms

This system was criticized by the maintenance experts on the following points :

(1) no distinction between tests and repairs : they were condidered together as operations by the optimization strategy,

(2) application of the single failure hypothesis,

(3) difficulties encountered by the expert in providing the elements of the diagnosis : tests and diagnosis rules.

1.3. Current work

The FLAG 2 system under development is meant to answer these criticisms.

In this paper we focus on the problem of building the diagnosis rules. This required considerable efforts from the expert in compiling knowledge spread throughout the wiring diagram and maintenance documents. Therefore only a small part of the NS was represented in the previous system.

To address this problem we choose to free the expert from this burden, providing him instead with the way to describe the elements of the NS and related knowledge through the use of a Knowledge Acquisition System (Fig. 1). This KAS allows a structural and functional description of the NS from which test interpretations are generated. Such an interpretation is a set of implications of the following form :

(f1) <single test result> and <conditions> ⟹ <conclusions>

2. The KAS

2.1. Structural and functional description

The expert describes the NS in a hierarchical and structural way as long as he possesses such knowledge, after which he decomposes it in a functional way.

* This research has been supported by DRET (French Defense Ministry) and has been worked on with Avions Marcel Dassault - Breguet Aviation

This description maps into the building of a net of black boxes linked by causal dependencies. Boxes originate through either the structural description of a component or a functional one.

Causal dependencies are flagged by contexts that determine their existence or not. In our case, some contexts are determined by selecting modes of devices. Modes are the different functioning setups in a component. Figure 2 shows a relay with its structural and functional representation and the resulting three dependencies ; one is permanent, the others are respectively set in the "ON" and "OFF" relay modes.

These dependencies may possess a type. In our application, GF-dependencies are those existing between components in the state of Good Functionning (GF) ; actually, the structural links between devices map to dependencies of this type. BF-dependencies (Bad Functionning dependencies) are originated in the knowledge of *failures causing irregular dependencies. BF-related flagging contexts may be set on the condition that some* components have not been discarded from the set of suspect components.

When setting a context relies on local conditions on input or output values at component terminals, a propagation model in the equipment may free the expert from expressing such knowledge as :

<knowledge of the global state of the equipment>$\Rightarrow$*<selected contexts>*

In our case, selecting any context of this type is equivalent to selecting component modes. Therefore the model could be restricted to the propagation of mode controls. In fact it is larger as the expert knows local transfer functions enabling the computation of outputs from inputs. For the relay (fig 2) we know that : if e3 = 28Volt then the "ON" mode is selected ; otherwise the "OFF" mode is selected. Propagating a value for e3 will enable this selection.
Knowledge of possible outputs for component terminals, without assuming good functioning, is also part of this model.

2.2. Diagnosis knowledge

Three kinds of diagnosis knowledge are expressed by the expert :

knowledge concerning the definition of the units that support the diagnostic (the "grain" of diagnostic) and how they are positioned in the structural and functional description. This definition is either implicit or explicit. It is implicit because the diagnosis decomposition matches with the structural and functional one. It is explicit when the decomposition is made only for diagnosis purposes. Analogical devices, for example, are often decomposed in several modules representing characteristic test values.

(2) knowledge used for suspecting components on which the interpretation of false test relies. A component will be suspect :
   (a) if it takes part, in GF, to the production of the tested output,
   (b) if its failure may explain the bad functioning of a component suspected according to (a).

(3) knowledge used to validate a component on which interpretation of true test relies. Knowing that a component output is correct the problem is to express :
   (a) which parts of the component are known to be active,
   (b) which parts may be validated,
   (c) which inputs are known to be correct.
The knowledge (c) provides a way of propagating backwards the symbolic value "ok" [5] and sometimes is assorted with an inverse transfer function enabling the propagation of a numerical value. The process of validation may proceed backwards as long as it is possible to propagate "ok". In the case of the relay, "high" is selected if s (fig 3) $\neq e_2$ and $s \neq e_3$ and consequently e_1 = "ok" (for "low" respectively $s \neq e_1$ and $s \neq e_3$ and consequently e_2 = "ok"). "Common" is selected in every case on condition that "low and "high" have already been validated. Moreover if "high" is selected we know that e_3 = 28Volts (respectively for "low" we know that $e_3 \neq$ 28Volts).

2.3. Additional diagnosis knowledge

The expression of failures associations provides pointers to cases of multiple failures known by the expert. They will be used by the repairing system and are not involved in the compiling process leading to *f* l forms.

3. Aided test interpretation

The KAS provides a tool for helping the expert in building test interpretations. Their form will further allow their integration in a net of non-monotonic data dependencies that is the skeleton of the diagnosis system.

3.1. False test

The interpretation of false test proceeds as follows :

(1) When an observed output is bad, the set E_{GF} of the components that participate in GF to the production of this output must be suspected. This set is built going back through the active GF-dependencies starting from the output point. They are found by selecting the associated context propagating mode controls in the equipment.

(2) The set S of suspected components is made up of those whose failure can explain the improper functioning of E_{GF} components. Obviously, we have both $E_{GF} \subset S$ and those components which can explain a BF by coupling. The later may be suspected given the non-validity of certain other components, and determined by taking BF-dependencies into account.

Thus the interpretation is built as a set of implications :

$$(f2)\left\{\begin{array}{l}\textit{<false test>} \Longrightarrow \textit{<suspected components>}\\ \textit{<false test> and <conditions>} \Longrightarrow \textit{<suspected components>}\\ \dots\end{array}\right.$$

The expert can then effect an analysis by withdrawing or adding suspected components, or even by validating some.

3.2. True test

The basis of the *interpretation of a true test lies in* the validation knowledge expressed by the expert. Various interpretations are possible and must be determined interactively. A true test observed at s may give rise to many interpretations. Concerning the relay we have seen that "common" validation is dependent on the validity of "high" and "low", and that their validation relies on conditions that deal with the output s and the *inputs* e_1 and e_2.

Let us assume that the conditions $s \neq e_1$, $s \neq e_2$ and $s \neq e_3$ cannot be evaluated because knowledge of the possible output values are not available prior to input e_1.

Various interpretations can thus be *envisaged* :

(1) No additional hypothesis is added in relation to the test specifications. In such a case, no component can be validated and it is not possible to continue the validation by "working backwards" through the circuit.

(2) If *we hypothesize that* H1 *and* L1 *are* valid, then it is possible to validate "common".

$$\text{(VALID H1) and (VALID L1)} \Longrightarrow \text{(VALID Com1)}$$

Consequently e_3 "ok" ; it is now possible to work *back towards the relay 2* where nothing, however, can be accomplished without making another hypothesis.

(3) Let us assume that the possible output value are available prior to e_3 and that we make the hypothesis that the set of modules belonging to Xe1 are valid. Eventually a value for e_2 may be computed. Consequently if $s \neq e_1$ and $s \neq e_3$ we will know that L1 is valid and working backwards through Xe2 with e_2 "OK" (and e_2 s) will be possible.
In this case we would have at least :

$$\sum_{c_i \in Xe1} \textit{(VALID } c_i\textit{)} \Longrightarrow \textit{(VALID L1)}$$

(4) Let us add, to the previous case, the hypothesis that H1 is valid. In this case we realise that Com1 is valid and that it is possible to work backwards starting from e_3 with a control value of the mode of L1 known to be active. *This allows* the validation of H2, L2 and Com2. Consequently :

$$\sum_{c_i \in Xe1} \textit{(VALID } c_i\textit{) and (VALID H1)} \Longrightarrow \textit{(VALID L1) and (VALID H2) and (VALID L2) and (VALID Com2)}$$

During the repairing session certain modules might be validated and the preceeding instances of implications would trigger.

We realise from this example that a great deal of conditional interpretations are possible. Many of them would not be of any use in that either they assume the validity of a great number of components (imagine several *hundred modules comprising* Xe1) or *no* test capable of making the conditions true exists. The choice of useful interpretation is left to the expert. The resulting test interpretation will have the following form :

$$(f3)\left\{\begin{array}{l}\textit{<true test>} \Longrightarrow \textit{<valid components>}\\ \textit{<true test> and <conditions>} \Longrightarrow \textit{<valid components>}\\ \dots\end{array}\right.$$

4. Contributions of KAS and conclusions

As for application, contribution of KAS is very positive :

(1) It allows the expert to express his entire knowledge concerning the equipment.

(2) *His knowledge is better exploited. Conditional test interpretations (f1 forms)* have the same expressive power than diagnosis rules relying in test results conjuncts *(f0 forms)*. However, building every interesting such conjonctive form revealed unfeasible, and led the expert to suspect more components than necessary.

(3) The developed representation scheme *enables other aides to be created,* notably those concerned with test generation.

We believe the KAS is general enough to be applied to other fields having such similar characteristics as :
. There is no well-established physical model as a descriptive basis that allows one to reason from "first principles" [3]. The interactions between LRU's are generally electrical but can be mechanical, even thermal.
. There is a close relation between structural and functional description of the equipment and the description of the "grain" of the diagnostic. In fact, the later is embedded in the former. As these two aspects are so interwoven, the expert remains indipensable to describe the material.
. The diagnosis knowledge can be compiled prior to operating *the repairing system. This approach* is not always feasible particularly in domains where a strong need for hierarchical diagnostic reasoning would imply an unpractical volume of compiled knowledge, as every [illegible]archical decomposing depending on diagnostic contexts would have to be envisaged.

REFERENCES

[1]R. CANTONE and al. "Model-Based Probalistic Reasoning for Electronics Troubleshooting" Proceedings IJCAI-83, Karlsruhe, Aug. 1983.

[2]J.M. DAVID & Y. DESCOTTE "Cost-efficient Troubleshooting" ECAI Conf. [illegible]

[3]R. DAVIS "Expert Systems : Where are we ? And where do we go from here ?" The AI Magazine : [illegible]

[4]M.R. GENESERETH "Diagnosis Using Hierarchical Design Models" AAAI 1982.

[5]W. HAMSCHER "Using Structural and Functional Information in Diagnostic Design" Technical Report N [illegible] MIT June 1983.

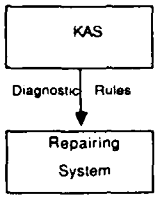


Figure 1

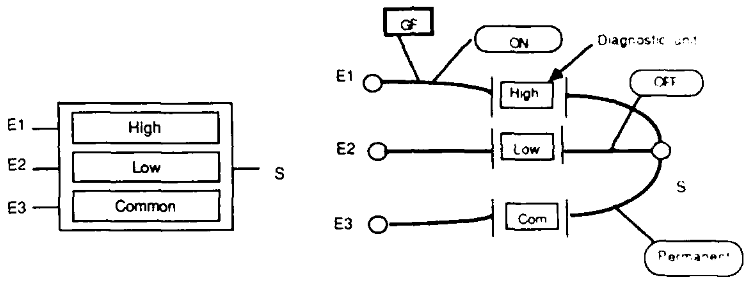


Figure2 . On the left: stuctural and functional description of the relay on the right the 3 dependencies with their $\boxed{\text{type}}$ and (context) and the different diagnostic units

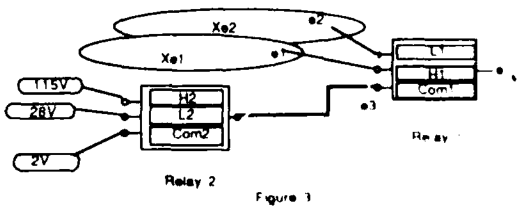


Figure 3

A REVIEW OF THE KNOWLEDGE ENGINEERING PROCESS

by
MICHAEL W. BIRD, PhD
Engineering Fellow
Lear Siegler, Inc./Instrument and Avionic Systems Division
4141 Eastern Avenue, S.E.
Grand Rapids, Michigan 49518-8727 USA

SUMMARY

Expert Systems are being applied in many areas of business, finance, medicine, and engineering and have the potential of being embedded in guidance and control systems. The key to development of an Expert System is the development of its knowledge base. A fundamental premise of Expert Systems is that knowledge about how to solve problems in the problem domain of the Expert System is available. Knowledge engineering is the process of acquiring this knowledge and incorporating the knowledge into the Expert System. The knowledge engineering process is divided into the following parts: acquisition of the knowledge, implementation of the knowledge using computer tools, testing of the knowledge base, and revision of the knowledge base based on the test results. This paper provides an overview of all parts of the knowledge engineering process. The basic types of data structures used for representing knowledge in an Expert System are described, including production rules and frames. The computer-based tools available for assisting engineers in acquiring and testing knowledge bases are discussed. This includes describing the general capabilities and features of the different types of Expert System development tools. The paper concludes by discussing the validation of Expert Systems - an issue critical to guidance and control applications.

BACKGROUND

Expert System concepts which were originally developed for tasks in medical diagnoses [1]*, geological data interpretation [2], and configuring computer systems [3] are now being applied in areas of engineering. Examples of engineering applications, which have been described over the past few years in conferences and journals, areintegrated manufacturing, automated VLSI layouts, maintenance diagnositcs, data fusion, battlefield management, autonomous land vehicle navigation, and pilot decision aiding. In addition to these engineering applications, this Artificial Intelligence technology has application to aircraft guidance and control systems, specifically flight control [4], route planning [5], and integration with the crew station [6,7]. Expert Systems embedded in these systems have the potential of increasing the flexibility, adaptiveness, and operational range of these systems.

A major distinction between an Expert System and today's conventionally structured computer program is the Expert System knowledge base. One of the early reports [8] on the concept of an Expert System described the role of knowledge in an Expert System as follows:

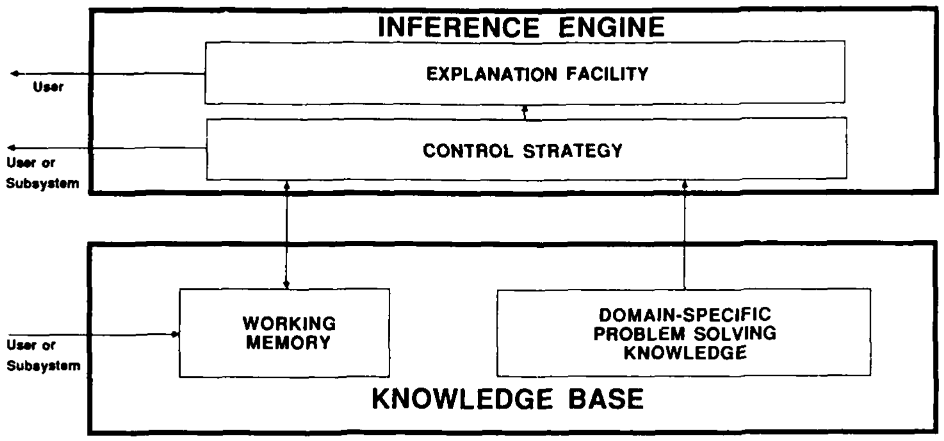
> "An Expert System is an intelligent computer program that uses knowledge and inference procedures to solve problems that are difficult enough to require significant human expertise for their solution. The knowledge necessary to perform at such a level, plus the inference procedures used, can be thought of as a model of the expertise of the best practitioners of the field.
>
> The knowledge of an Expert System consists of facts and heuristics. The facts constitute a body of information that is widely shared, publicly available, and generally agreed upon by experts in a field. The heuristics are mostly private, little discussed rules of good judgement (rules of plausible reasoning, rules of good guessing) that characterize expert-level decision making n the field. The performance level of an Expert System is primarily a function of the size and quality of the knowledge base that it possesses."

Because the knowledge base is the essential element of an Expert System, these systems are also referred to as Knowledge Based Systems.

The basic structure of an Expert System is shown in Figure 1. The structure is divided into two fundamental parts - a knowledge base and the inference engine. The knowledge base contains the facts, heuristics, and problem solving rules for the domain of problems being addressed by the system. The inference engine is the procedure for using the knowledge base in solving a particular problem. The knowledge base is unique to a specific problem domain, but the inference engine may be common to a number of domains with similar characteristics.

At the next level of detail, an Expert System has the four blocks shown in Figure

* [illegible] references at end of paper.



GENERAL STRUCTURE OF EXPERT SYSTEM

FIGURE 1

Working Memory contains the declarative knowledge about the particular problem being solved and the current state of affairs in the attempt to solve the problem. There are several ways to represent data in working memory: predicate logic, frames, and semantic networks. These representations are described below.

Domain-Specific Problem Solving Knowledge contains relationships between several pieces of data in working memory. These relationships establish how the data can be manipulated to solve problems in the domain. The most common form of relationship is the production rule which also is defined below. In the knowledge base, the rules represent domain facts and heuristics - judgements of good actions to take when specifics of the problem arise.

Control Strategy makes decisions about how to use the domain-specific problem solving knowledge to solve the particular problem defined by data in working memory. For production rules, the control strategy is implemented with an interpreter that decides how to apply the rules to infer new knowledge, i.e., new data for the working memory, and a scheduler that decides the order in which the rules should be applied.

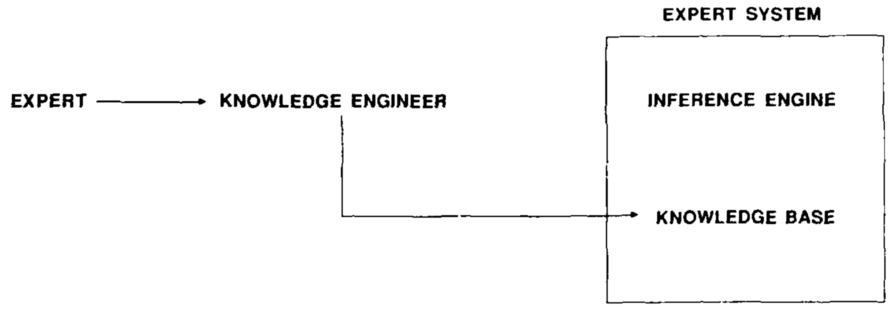
Explanation Facility involves explaining to the system user the line of reasoning that led to the problem solution. With a production rule knowledge base, this is often accomplished by retracting the chain of production rules that led to the solution and translating them into some form readily intelligible to the user.

KNOWLEDGE ENGINEERING APPROACHES

The process of developing or building an Expert System is called knowledge engineering. This process involves obtaining problem solving expertise, building a knowledge base from this expertise, and implementing the knowledge in an Expert System. The expertise in solving problems in the domain of the system can come from a number of sources: one or more human experts, books and manuals, or simulations. A majority of the research and applications work devoted to building Expert Systems has utilized human experts, usually one expert, as the problem solving knowledge source. This can be verified by reviewing the Artificial Intelligence periodicals and conference proceedings over the past decade. In addition to human expertise, material in texts and reference manuals is often used to complement the human expertise or to aid in acquiring knowledge from the human expert. For some complex problems where limited human expertise exists, detailed and extensive simulations of solutions can provide insights into "rules of thumb" and heuristics for the problem solving knowledge. This paper concentrates on reviewing knowledge engineering approaches that utilize human experts as knowledge sources, primarily because these approaches have been emphasized in the knowledge engineering literature.

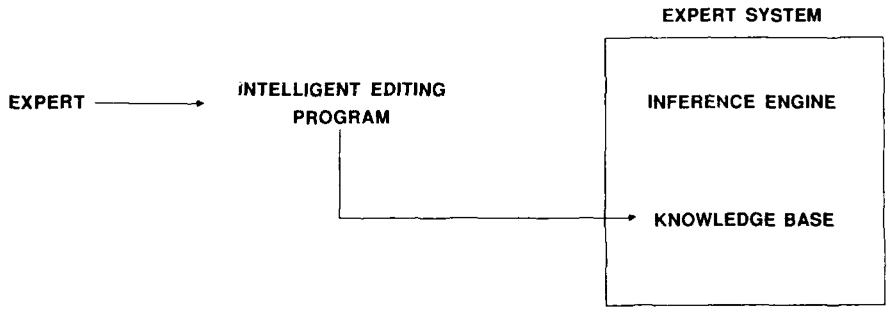
The key component of the knowledge engineering process is knowledge acquisition. This is the process of transferring and transforming the problem solving expertise from the knowledge source to the computer program embodying the Expert System. This process has proven to be difficult and has caused knowledge acquisition to be identified as a bottleneck in the construction of Expert Systems [1,8].

Currently, there are two methods for knowledge acquisition which are illustrated in Figure 2 [1]. In the first method (Figure 2a), the expert converses with a second person, the knowledge engineer, who extracts the expert's problem solving expertise. In this method, the knowledge engineer becomes very knowledgeable about the way the expert solves problems, determines how to best represent the knowledge in the computer, and uses this representation to develop the knowledge base and system. The second method (*Figure 2b) is described in Reference 1 as follows:* "*The expert conversant with computer technology can interact more directly with the Expert System via an intelligent* editing program. Here, the expert converses with the editing program rather than with a knowledge engineer. The editing program must have sophisticated dialogue capabilities and considerable knowledge about the structure of knowledge bases. This method replaces one set of communication problems (expert to knowledge engineer) with another (expert to program)."



KNOWLEDGE ENGINEERING VIA KNOWLEDGE ENGINEER

FIGURE 2a



KNOWLEDGE ENGINEERING VIA AN INTELLIGENT EDITING PROGRAM

FIGURE 2b

This paper focuses on the knowledge acquisition process. The various data structures used for knowledge representation are described. Then, the major stages of the knowledge acquisition process are discussed. These sections are followed by descriptions of the types of software tools available for acquiring knowledge and developing Expert Systems.

KNOWLEDGE REPRESENTATIONS

The knowledge utilized by an Expert System typically has several forms. There will be declarative knowledge which may include definitions of terms used in the problems being solved by the Expert System (e.g., "actuator", "aircraft attitude", "waypoints") *and descriptions of objects and their relationships to each other (e.g.,* "MP is a mission plan with targets T_1 and T_2"). There will be problem solving knowledge (e.g., "If lateral path deviation is greater than one mile and next waypoint is less than five miles, then go direct to next waypoint").

Artificial Intelligence researchers have applied the Expert System concept to a variety of problems and have established a number of ways to represent knowledge [9]. Experience has shown that no representation by itself is efficient for all types of knowledge. Some representations are better suited for declarative knowledge and others for problem solving knowledge. First order predicate logic, semantic networks, and frames are the most widely used representations for declarative knowledge while production rules predominate in the representation of problem solving knowledge. A combination of production rules and frames has been used recently in a number of Expert System development tools to provide the flexibility needed for representing knowledge in a wide variety of applications [10,11].

This section describes four ways of representing knowledge. This includes descriptions of their data structures and the advantages and disadvantages of each representation.

First Order Predicate Logic

One of the earliest areas of research in Artificial Intelligence (AI) was in theorem proving which developed first order predicate logic. In this representation, facts and relationships are represented as predicates, for example, "Caesar was a ruler" and "A Roman was a person" are represented as "ruler (Caesar)" and "for all x, Roman (x) > person (x)".

First order logic is appealing because there are formal, theorem-proving type procedures for deducing conclusions from the set of predicates. The resolution principle [12] is a method for making deductions that is most often associated with First Order Logic. This method has given rise to many of the ideas behind the AI programming language PROLOG.

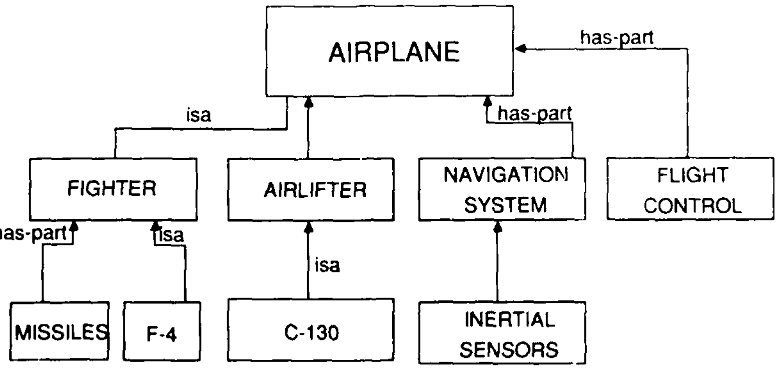
There are several reasons that first order predicate logic is a useful representation. Expressing facts in logic is the most natural way for some applications. Logic is precise - with formal methods available for drawing conclusions from the facts. Logic is flexible, has well-defined semantics, allowing facts to be expressed in a simple way without having to consider their possible use. Logic is modular - new facts can be incrementally added to the knowledge base without adversely effecting the deductions of the system.

One disadvantage of first order logic is the fine grained structure of its representation making it difficult to define complex objects and relationships and difficult for the domain experts and knowledge engineers to understand. A second disadvantage is difficulties in combining logic-based declarative knowledge with heuristics needed for problem solving. This is especially acute in problem domains involving large combinatoric search processes.

Semantic Networks

This way of representing knowledge is based on a network structure. A semantic network consists of points called nodes which are connected by links called arcs. The arcs describe the relations between the nodes. Nodes stand for objects, concepts, or events. Two often used arcs for representing relationships are "isa" and "has-part" [9].

A simple example illustrating nodes and arcs of a semantic network is shown in Figure 3. Two of the facts shown in this example are, "The F-4 is a fighter" and "A fighter is an airplane". Because of the relations defined by the arcs, a third statement can be inferred from the network, "The F-4 is an airplane".



SIMPLE SEMANTIC NETWORK OF AN AIRPLANE

FIGURE 3

Relations such as "isa" and "has-part" establish a property inheritance hierarchy in the network. This means items lower in the network can inherit properties from objects higher up in the network. For example, in the example of Figure 3, the navigation system, flight control system, and inertial system are stored once at the aircraft level, but inherited at the lower levels for each type of aircraft. Use of inheritance arcs saves huge amounts of computer memory space.

Semantic networks are a very popular representation scheme. The node-and-arc structure match-up with the symbols and pointers used in symbolic computation and have a natural compatibility with the association relationships defined in the psychology studies of memory. Semantic networks are a useful way to represent knowledge for problem domains that have well-established taxonomies. This is why semantic networks have been successfully used in natural language research [13]. Difficulties arise when semantic networks are applied to large, complex problems where there are usually problems in interpreting the semantics of the network structure [9]. These problems are reduced through aggregated network structures like frames which are discussed next.

Frames

A frame is a data structure for representing an object or classes of objects. The data structure has fields, usually called slots, used to describe the attributes of the object. Some frames in a frame system are used to provide generic descriptions of objects. For example, a generic airplane frame may have slots for attributes such as the number of engines, cabin configuration, cockpit layout, and physical dimensions. A frame for a particular airplane has the same slots - inherited from the generic frame with the contents of the slots fully specified.

A frame structure can have an inheritance mechanism similar to semantic networks. Specialized slots in a frame can establish a property inheritance hierarchy among frames, which allows information about a parent frame to be inherited by its children. This is similar to the inheritance relationship in an "isa" arc of a semantic network. With this inheritance feature, a system of frames can be organized much like a semantic network where each node in the network is a frame, the topmost nodes representing general concepts, and the lower nodes being specific instances of these frames. In many frame systems, a slot can have subslots of its own, i.e., a slot has a frame structure.

The attributes described in the slots and the property inheritance hierarchy of a frame represent static facts about the object or classes of objects. Frames can also have a dynamic or procedural behavior which is achieved by attaching computer procedures to slots in the frames. These procedures are referred to as attached procedures. In general, there are two types of attached procedures [9]: an *If-Needed procedure* that executes when details need to be filled-in about the slot and an If-Added procedure that executes when the value of a slot has changed.

Major advantages of a frame system are the following: information for each object or class of objects is located together, accessing and manipulating the information is easier, the inheritance feature minimizes duplication of information about objects in the knowledge base, and values for the object's attributes are created when needed. The main disadvantage of frames is that, except for attached procedures, they have no inherent capability for utilizing their data for problem solving.

Production Rules

A production rule consists of an IF part and a THEN part. The IF part [illegible] set of conditions under which the actions in the THEN part of the rule are to be taken. In general, the rule structure is as follows:

IF Condition 1, Condition 2, ..., Condition n

THEN Action 1, Action 2, ..., Action m

where the conditions and actions are patterns of symbols [illegible]

In the typical Expert System, the rules making up the knowledge base [illegible] problem solving knowledge. There are two methods an inference engine [illegible] cessing the rules: forward chaining and backward chaining.

In forward chaining, the inference engine compares the [illegible] the conditions of the various rules and tries to fire, i.e., execute [illegible] rule whose conditions are matched. If the conditions [illegible] a conflict resolution method is utilized to determine what rule [illegible] fires, the facts associated with its actions are added [illegible] potentially trigger other rules. This method is often called [illegible] because the rule conditions are being matched against the facts.

Forward chaining is very useful in real time applications. [illegible] the rules can represent event/response type knowledge.

In backward chaining, the system starts with a hypothesized [illegible] part of the rules, and finds if one of those actions includes the goal. [illegible] takes the conditions of the applicable rule, makes them subgoals, [illegible] whose actions would satisfy these subgoals. The process repeats [illegible] and subgoals match the known facts. This method is called goal-driven reasoning.

Of the current methods of representing knowledge, rules are the most widely used way for representing problem-solving knowledge. They are often thought of as relatively independent pieces (or chunks) of know-how. This is the type of know-how psychologists feel humans use for perception and thinking. Rules have been effective for representing the following types of knowledge experts use in solving problems: preferred problem-solving tactics, plausible intermediate steps to take in finding solutions, and just plain shortcuts found from experience:

Some of the reasons production rules are effective in knowledge representation are the following [9]:

- Modularity - rules can be added or deleted without changing the other rules in the knowledge base
- Uniformity - because of their rigid structure, rules can be easily understood by other people besides the expert and knowledge engineer and they can be efficiently processed by the inference engine
- Naturalness - the rule format is an easy way for a human to express certain types of knowledge, in particular what to do in predetermined situations

The modularity and independence features of rules, which make them attractive for particular types of problem solving knowledge, make rules less efficient for representing objects, their attributes, and their static relationships - representations handled efficiently by frames. In a rule system, the representation of an object or object attributes is distributed over a number of rules with dependencies between rules. The data structure that can be naturally associated with objects is lost when compiled in the form of production rules.

The complementary nature of production rules and frames has been recognized and is being used to increase the effectiveness of Expert Systems. A later section of this paper discusses how some computer programs designed as tools for the knowledge acquisition are combining frames and production rules for hybrid representations. Frames represent objects and their relationships that are referred over rules. This simplifies the rule sets in exchange for the added complexity of a frame system and the increased complexity of the inference engine which must process frame data as well as rules and facts.

THE MAJOR STAGES OF KNOWLEDGE ACQUISITION

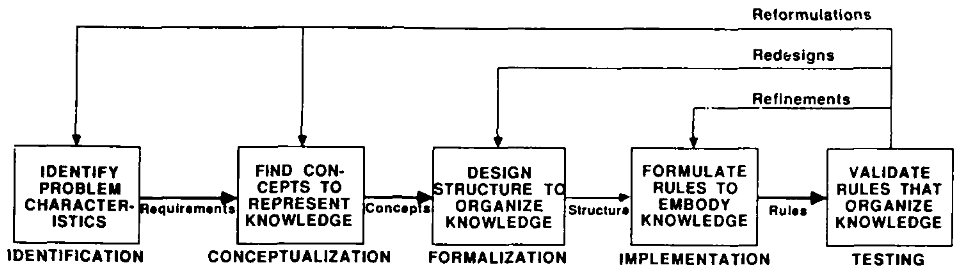
There is no established, well-defined process for developing the knowledge base for an Expert System. Since the mid-1970's, when the concepts for Expert Systems were formulated as part of research into Artificial Intelligence methods, there has been considerable study of knowledge acquisition processes [1,14,15], but only guidelines are available defining a knowledge acquisition methodology. Guidelines have evolved from work ranging from theoretical studies to empirical system development and these guidelines are in general agreement. They agree that the knowledge acquisition process has a number of phases, or stages, on what the stages should be and, in general, on what should be accomplished in each stage. This section will describe these stages of the knowledge acquisition process, drawing heavily on material in References 1 and 16.

The knowledge acquisition process can be viewed as five highly interdependent and overlapping stages: identification, conceptualization, formalization, implementation, and testing. Figure 4, which is borrowed from Reference 1, illustrates these stages and how they interact. In general, the process is incremental - acquiring knowledge for solving simple problems in the Expert System's domain and expanding this knowledge to solve harder problems in the domain. The incremental approach involves building a prototype knowledge base and using some form of Expert System development to test, evaluate, and refine the knowledge base.

Identification Stage

The initial step in acquiring knowledge is establishing the important features of the classes of problems that will be solved by the Expert System. This involves identifying the domain expert (or experts) who will participate in the knowledge acquisition process. The knowledge engineer works with the domain expert during this initial stage to adequately define the problem domain so development of the knowledge base can begin.

Problem identification includes identifying the type and scope of problems being solved. An informal characterization of the terms describing the problems, key concepts, and how to divide the problem into subproblems is established. The goals or objectives of the Expert System are defined. Examples of Expert System goals are the following: 1) capturing the expertise of a key expert in the organization, for example, a guidance system designer, gyro technician, or flight control system maintenance expert, 2) managing the data displayed on the cockpit displays, and 3) pruning the search space of a trajectory optimization algorithm.


Reformulations
Redesigns
Refinements
IDENTIFY PROBLEM CHARACTER-ISTICS
Requirements
FIND CON-CEPTS TO REPRESENT KNOWLEDGE
Concepts
DESIGN STRUCTURE TO ORGANIZE KNOWLEDGE
Structure
FORMULATE RULES TO EMBODY KNOWLEDGE
Rules
VALIDATE RULES THAT ORGANIZE KNOWLEDGE
IDENTIFICATION
CONCEPTUALIZATION
FORMALIZATION
IMPLEMENTATION
TESTING

STAGES OF KNOWLEDGE ACQUISITION

FICURE 4

Conceptualization Stage

During conceptualization, the knowledge engineer and domain expert decide what kinds of concepts and relations are needed to describe problem solving ideas in the domain. Conceptualization includes exploring problem solving strategies, dividing the problems into subtasks, and characterizing problem constraints.

This stage involves extensive interactions between the knowledge engineer and expert and develops the essential elements of the knowledge base. The interactions with the expert must include specific examples of solving prototype problems. Getting an expert to provide accurate verbal descriptions (protocols) of his own personal problem solving process is difficult. Some Artificial Intelligence researchers recommend some form of organization be employed in the knowledge engineer - domain expert(s) interviews, such as a thinking aloud protocol and related methods from the area of problem solving psychology [17].

The knowledge engineers may find it useful or necessary to diagram or organize the concepts and relationships arising from the interactions with the expert. Producing a "paper knowledge base" consisting of English sentences representative of the concepts, relationships, and problem solving strategies from the protocols recorded during the interviews is part of one organization's knowledge acquisition methodology [15]. A paper knowledge base starts making the knowledge structure clear during the conceptualization stage. It can also aid in establishing the granularity needed in the knowledge base, i.e., to what level of detail the knowledge needs to be represented to solve problems in the domain.

Formalization Stage

The formalization process involves mapping the major concepts, relations, sub-problems, information flow, and problem solving ideas into a formal knowledge representation framework. The knowledge engineer takes a more active role in the interactions with the domain expert(s), describing to him/her the knowledge structures and problem solving paradigms that appear compatible with the problems in this domain. As progress is being made in solidfying a knowledge representation framework, selection of an Expert System development tool with a knowledge representation architecture compatible with the framework is selected. If a development tool was already selected, because of project schedule constraints, development of the knowledge representation framework takes into account the representation capabilities of the selected tool. The output of the formalization stage is a set of partial specifications describing how the knowledge base can be represented within the chosen Expert System development tool.

There is disagreement among Artificial Intelligence practitioners on how elaborate an initial knowledge base framework is required before starting to prototype on an Expert System development tool. Many advocate quickly putting together a preliminary concept of the knowledge base, implementing it in a prototype system, and shaping and determining a good knowledge base framework and structure through what is learned from the expert interacting with the prototype [16]. Others advocate providing more structure in the formalization activity before prototyping. These advocates feel the implementation constraints present in a prototype system will prevent convergence on an efficient knowledge base structure if commitment to prototyping is premature.

Methodologies designed to place structure in the knowledge base formulation process concentrate on uncovering the underlying knowledge base structure, structure of the problem solving knowledge, domain objects, relationships between objects, and constraints of the problem domain. These methodologies typically involve mapping the verbal data collected during the sessions with the expert into an implementation-independent description of the domain knowledge [18,19]. The descriptions are high-level, abstract representations of the problem's knowledge structure and are distinctly different from the knowledge representation languages (frames, rules, etc.) discussed above. The descriptions consist of typologies of basic elements and structuring relationships for certain classes of porblem solving tasks. The basic elements will typically be objects, relationships, dynamic operations, and knowledge structures that guide the selection and use of these operations [19]. The formalization process consists of repeated cycles of elicitation and analysis of verbal data from the expert aimed at refining the top-level description of the knowledge base.

The decision on whether to prototype as soon as possible or to utilize some form of structured method for formalizing a knowledge base structure before prototyping depends on the project objective and duration. If the objective is to show the feasibility and performance of an Expert System for the problem domain and with a limited project budget, prototyping as soon as possible is recommended. A structured approach for determining the knowledge representation framework takes time and may contribute little to demonstrating feasibility and projected performance. If the objective is the complete development of an Expert System, a structured approach to developing a sound knowledge representation framework for the problem domain is recommended. A good understanding of the knowledge representation framework is essential for the knowledge base maintenance activities required for validating the Expert System and later for updating it with new knowledge during its operational use [20].

Implementation Stage

The implementation process turns the formalized knowledge into a working computer program. The domain knowledge made explicit during the formulation stage is implemented on the Expert System development tool using the knowledge representation capabilities of the tool. Implementation includes integrating the operations of the different parts of the prototype Expert System, i.e., working memory, problem solving knowledge base, inference engine, and explanation facilities so the system functions properly for testing.

The implementation stage should proceed rapidly, especially for a feasibility project, because the objective is to check the effectiveness of the concepts and knowledge representation structure developed during the conceptualization and formulation stages. The implementation will change as the testing of the prototype progresses.

The variety of Expert System development tools that can be used for the implementation and testing stages are discussed in the next section.

Testing Stage

The testing stage involves evaluating the knowledge base and the knowledge representation framework using the Expert System development tool. The domain expert evaluates the performance of the prototype system and helps the knowledge engineer revise the system. The system is tested with problems that cover the domain including prototypical cases and hard cases expected to probe the domain boundaries.

Many revisions of the knowledge base part of the system are expected during testing to attain the performance expected by the expert. There may also be modifications to the inference engine; although, these are expected less often. Revisions may require revisiting the earlier stages of the knowledge acquisition process. Typically, the revisions will occur in the following order:

1. Refinement of the prototype system by adjusting production rules, working memory characterizations, and the inference engine's control mechanism until the knowledge base gives the expected performance.

2. If refinement doesn't converge to acceptable performance, redesign of the knowledge representation framework developed during the formulation stage and acquisition of additional or new knowledge through further expert - knowledge engineer interviews.

3. If the difficulties are so serious that knowledge base redesign and prototype refinements are not adequate, serious mistakes in identifying or conceptualizing the problem have occurred. It may be necessary to rescope the problem, provide additional data or information to the system, or re-evaluate the system goals or objectives.

In addition to evaluating the performance of the knowledge, the utility of the Expert System built around the knowledge base is evaluated. The interface to the user, whether a human user or another system, is evaluated for efficiency, i.e., are the system solutions organized, ordered, and presented at the right level of detail? Also, the time for computing solutions is assessed to determine if the system will be fast enough to satisfy the user. The assessment can be actual measurements of solution times, if the prototyping computer system will be the operational system; otherwise, the solution time is estimated based on the processing architecture of the operational hardware. If the solution times do not meet user requirements, there are a number of options: more efficient implementations of the knowledge base and inference engine, use of a traditional procedure-oriented programming language such as Ada or FORTRAN for implementation or reformulating the knowledge representation structure and acquiring more problem solving heuristics from the expert to reduce the amount of computations required of the inference engine to solve problems.

EXPERT SYSTEM TOOLS

Expert System tools are programming systems that aid in the task of implementing and testing the knowledge elicited from the domain expert or experts. These programming systems range from very high-level programming languages to low-level support facilities and can be divided into four categories [1,16]: programming languages, knowledge engineering languages, system building aids, and support facilities.

Programming Languages

The programming languages generally used for Expert System applications are symbol-manipulation languages such as LISP and PROLOG. The LISP language is built around mechanisms for manipulating symbols in the form of list structures. These structures are useful building blocks for representing abstract and complex concepts needed in an Expert System. LISP is the most widely used programming language for Expert System applications; although, PROLOG, which was developed in the mid-1970's, is gaining in popularity. The PROLOG (PROgramming in LOGic) language is usually based on a theorem prover method, which can tell if a given set of logical formulas has any contradictions or not.

A programming language offers the most flexibility for developing an Expert System and implementing a knowledge base; however, there are major disadvantages. An extensive amount of time and manpower is required to develop a complete Expert System that has production rules, an inference engine, and adequate explanation and debug capabilities. If a mixture of representation structures are needed, for example, both frames and production rules, the development effort is even more extensive. This kind of development will be very inefficient early in a knowledge acquisition project, especially if it is used as the prototype Expert System. Significantly less effort will be expended if a knowledge engineering language offering a choice of representation facilities is used for mechanizing the prototype Expert System.

Knowledge Engineering Languages

Knowledge engineering languages are languages designed specifically for developing and debugging Expert Systems. They have special facilities that ease the development of Expert Systems, but are less flexible than ordinary programming languages. The computer programs that embody these languages are usually written in one of the two Artificial Intelligence programming languages, LISP or PROLOG, and are hosted on computers that process these languages. Recently, some of the knowledge engineering languages have been written in conventional languages, such as C, for hosting on minicomputers and workstations.

At a minimum, a knowledge engineering language provides the means for representing the various parts of a knowledge base, i.e., objects, relations, and problem solving knowledge. In addition, the Expert System components needed for processing the knowledge base representations are included in the language. A knowledge engineering language has the capabilities a knowledge engineer needs to develop a total Expert System. Reference 22 surveys the variety of knowledge engineering languages currently available.

Some knowledge engineering languages use one way for representing knowledge. OPS5 is a well-known and well-documented language based on production rules [21]. This language is a result of the extensive amount of Artificial Intelligence research on production rule systems. It has been successfully applied to many applications where production rules are an effective means for representing the knowledge base.

Recently, knowledge engineering languages have been developed that combine the features of the different knowledge representation techniques, i.e., they combine the features of frames, rules, and semantic networks. These languages provide the knowledge engineer a number of ways of representing knowledge and, therefore, are capable of developing Expert Systems for a wide range of problem domains. These general purpose languages have a number of choices for knowledge representation, but are usually restricted in the control mechanisms used in the Expert System inference engine. The more elaborate languages include a number of sophisticated AI techniques such as search strategies and knowledge base maintenance, making it easier for the knowledge engineer to implement different forms of problem solving strategies. Two examples of general purpose, knowledge engineering languages are ART (Automated Reasoning Tool) and KEE (Knowledge Engineering Environment) [22].

Another way the knowledge engineering languages ease the knowledge engineer's implementation and testing tasks are through specialized user interfaces. The more sophisticated languages have many of the following user interface features:

- access to several types of program information through multiple windows on the computer display,
- menus and a pointing device for easy system control,
- simultaneous viewing of evolving short term memory and production rule execution sequences during Expert System operation,
- graphical representation of hierarchical inheritance relationships in the frame system, and
- facilities to incrementally add, delete, and modify any knowledge base statement at any time

System-Building Aids

System-building aids are computer programs that help acquire and represent the domain expert's knowledge. These programs are designed for a particular problem domain, have knowledge representations tailored to that domain, and possess user interfaces that help elicit knowledge from the domain expert. These programs often are designed to interface directly with a domain expert.

Compared with programming languages and knowledge engineering languages, few system-building aids have been developed. Relatively complete system-building aids have been developed for fault diagnosis, classification, and configuring system problems [4,23,24]. The knowledge encoding in these computer programs is conceptually close to the actual expert knowledge and, hence, makes the knowledge base easier to construct, refine, and update. Maintenance of the knowledge base is also made simpler with these aids.

Support Facilities

Support facilities are extra software packages that come with the Expert System development tool making the tool easier to use, friendlier, and more efficient. In general, there are three different types of support facilities: editors, knowledge enhancement tools, and explanation facilities.

The most common editors are language editors with knowledge of the specific syntax of production rules and other knowledge representation structures. Editors are used interactively, with the editor prompting the knowledge engineer for portions of a rule or declaration, supplying defaults, correcting spelling, and performing type-checking as necessary. Editors remove the drudgery of discovering and correcting syntax errors. More sophisticated editors statically analyze the knowledge base entered by the knowledge engineer, analyzing new rules to help find undesirable interactions with existing rules.

Knowledge enhancement tools analyze the Expert System computer program dynamically during execution. Dynamic analyses include traces of rule firings and working memory modifications, summaries of rule execution statistics, and the ability to step forward and backward through the Expert System execution. The more sophisticated knowledge enhancement tools suggest areas in the knowledge base where the rules need to be generalized or specialized and areas where new rules are needed to fill problem solving knowledge gaps.

Explanation facilities attempt to answer the knowledge engineer's questions relating to Expert System behavior. The least sophisticated facility uses canned text linked to production rules or slots in frames. The next level of sophistication generates explanations based on traces of production rule executions or frame activities.

VALIDATION

The Expert System concept is making the transition from being a research idea over being a software engineering technology. Demonstrating feasibility and performance of Expert Systems for different applications has been emphasized during this transition. Little attention has been given to the reliability and robustness of Expert System software. There is concern that Expert Systems developed heuristically by trial and error methods will be inherently less reliable than conventional software developed from precise specifications. Some AI researchers feel the best Expert Systems can do is to imitate human experts, who are themselves imperfect [25].

An Expert System is still a computer program, especially when viewed from the perspective of software validation requirements. And, except for the separation of knowledge from control (the inference engine), an Expert System has the same characteristics as a conventional computer program. For example, both Expert Systems and conventional computer programs use heuristics and "rule of thumb" judgements. The difference is that conventional computer programs embed this type of knowledge in a procedural implementation.

Today's software engineering methodologies for developing reliable and robust computer programs can be applied to Expert System development. These methodologies typically divide the computer program development activities into phases: requirements analysis, specification, design and implementation, and test. The stages of the knowledge acquisition process discussed earlier in this paper correspond closely with these phases. This correspondence is even closer if the software engineering principles of modularity, layering, and information hiding are incorporated into the design of production-level knowledge bases and Expert Systems.

However, the testing of an Expert System is different from that of a conventional computer program. First, an Expert System often exhibits non-deterministic behavior because the pieces of the knowledge base (production rules and frames) strung together by the inference engine to solve problems are determined during system execution. This makes the system behavior unrepeatable for dynamic, time-varying problems and, therefore, makes the system difficult to debug. Second, there is no precise input-output relationship for production rules as there are for procedures in conventional computer programs. This makes it difficult to use input-output analysis for testing. Third, the number of ways production rules can be activated is too large to be able to realistically generate test cases covering every branch in the program.

Consequently, prototyping is the only effective way to test an Expert System. Experiences of the experts and users with the prototype can reveal vital problems in the design. However, this type of testing can present a problem because prototype testing of a large system can take substantial effort and time.

REFERENCES

1. F. Hayes-Roth, D. A. Waterman, and D. B. Lenat, Building Expert Systems, Addison-Wesley Publishing Company, Reading, Massachusetts, 1983.

2. G. Kahn and J. McDermott, "The Mud System", IEEE Expert, Spring 1986, pp. [illegible].

3. J. McDermott, "R1: A Rule-Based Configurer of Computer Systems", Carnegie-Mellon University, CMU-CS-80-119, April 1980.

4. J. Davidson, M. A. Allstadt, R. A. Bell, C. J. Pittman, [illegible], and M. [illegible] Zampi, "Expert Maintenance Diagnostic Systems for the F-15 Flight Control System", National Aerospace Electronics Conference, Dayton, Ohio, May 198[illegible], pp. [illegible].

5. J. F. Gilmore, "The Autonomous Helicopter System", Proceedings [illegible] Applications of Artificial Intelligence, Volume 485, May 1984, pp. [illegible].

6. R. Schudy, B. Wilcox, and R. Shu, "In Real-Time w[illegible]", [illegible] Annual Aerospace Applications of Artificial Intelligence Conference, [illegible] October 1986, pp. 62-66.

7. L. D. Pohlmann and J. R. Payne, "Pilot's Associate [illegible] Inside", National Aerospace Electronics Conference, May 198[illegible], pp. [illegible].

8. E. A. Feigenbaum, "Knowledge Engineering [illegible]", [illegible] Department, Stanford University, 1982.

9. A. Barr and E. A. Feigenbaum, The Handbook of Artificial Intelligence, [illegible] HeurisTech Press, Stanford, California, 1982.

10. R. Fikes and T. Kehler, "The Role of Frame-Based Representation [illegible]", Communications of the ACM, Volume [illegible], Number [illegible], September [illegible].

11. C. Williams, "Expert Systems, Knowledge Engineering, [illegible]", IEEE Expert, Volume 1, Number 4, Winter 198[illegible], pp. [illegible].

12. N. J. Nilsson, Principles of Artificial Intelligence, [illegible] 1980.

13. P. H. Winston, Artificial Intelligence, [illegible] Massachusetts, 1984.

14. F. Hayes-Roth, P. Klahr, and [illegible] Programming, and Knowledge Refinement [illegible] Santa Monica, California, May 198[illegible].

15. M. Freiling, J. Alexander, [illegible] Knowledge Engineering [illegible] 1985, pp. [illegible]-[illegible]4.

16. [illegible] Waterman, Guide to Expert Systems, [illegible]

17. V. R. Waldron, [illegible] Second Conference [illegible] Artificial Intelligence [illegible] December 198[illegible], pp. [illegible].

18. [illegible] Alexander, M. [illegible] [illegible]

19. [illegible]

21. L. Brownston, R. Farrell, E. Kant, and N. Martin, _Programming Expert Systems in OPS5 - An Introduction to Rule-Based Programming_, Addison-Wesley, Reading, Massachusetts, 1985.

22. J. F. Gilmore and K. Pulaski, "A Survey of Expert System Tools", Second Conference of Artificial Intelligence Applications, Miami Beach, Florida, December 1985, pp. 498-502.

23. L. Eshelman and J. McDermott, "MOLE: A Knowledge Acquisition Tool that Uses Its Head", AAAI-86, Volume 2, Philadelphia, Pennsylvania, August 1986, pp. 950-955.

24. H. Wu, H. W. Chun, and A. Mimo, "ISCS - A Tool for Constructing Knowledge-Based System Configurators", AAAI-86, Volume 2, Philadelphia, Pennsylvania, August 1986, pp. 1015-1021.

25. [illegible], "Towards a [illegible] Expert Systems", _IEEE Expert_, Summer 1986, pp. [illegible].

A RULE-BASED SYSTEM FOR ARRIVAL SEQUENCING AND SCHEDULING IN AIR TRAFFIC [illegible]

by
U. Volckers
Deutsche Forschungs- und Versuchsanstalt
für Luft- und Raumfahrt (DFVLR)
Institut fur Flugfuhrung
Flughafen
D-3300 Braunschweig, Germany

Summary

Monitoring, planning, conflict solving and decision making in air traffic [illegible] carried out mainly by human controllers. Computer assistance and in particular the advance [illegible] Intelligence (AI) now offer the possibility to support controllers in their intellectual [illegible] enhance performance and efficiency in air traffic control (ATC).

A rule-based planning system is presented, designed to assist human [illegible] merging high density inbound traffic into congested airports.

Topics to be discussed in more detail include: the rationale for the design [illegible] as well as the crucial requirements for the development of a rule-based system [illegible]

The rule-based system developed by DFVLR uses the OPS5 production system and is programmed [illegible] LISP with embedded "C" functions. It should be emphazised that the system still has limited [illegible] The considerations which led to the selection of suitable programming languages, knowledge representation, [illegible] processing structures etc. are discussed in detail. The limitations [illegible] improvements are outlined.

1. Introduction

Despite the introduction of advanced technical equipment in Air Traffic Control the task of the Air Traffic Control Officer (ATCO) has remained highly "conventional" or "manual".

With the first generation of automation in Air Traffic Control, automated systems for radar [illegible] processing and tracking, flight plan data processing and for communication were provided to controllers. However, the actual control process still had to be carried out in the brains of the human controllers.

In todays second/third generation of automation in ATC, advanced computer-systems are used for [illegible] information processing and planning functions. The ATC units are provided with filtered data, result [illegible] solutions, which assist the controllers in their decision making process (References [illegible]).

However the appropriate task sharing between human controllers and the computer and the distribution of responsibility between man and machine has raised many questions and problems, yet to be solved.

A solution, where the computer system works fully automatic, with the human controllers only monitoring the system, in order to take over in case of system failure, presently is for many reasons neither practicable nor acceptable.

That is the reason why so far the degree of automation is guided by the philosophy of giving computer assistance to the human controllers. The systems releave the controllers of routine functions and support [illegible] the necessary data to take safe, efficient and orderly control actions. The decision making itself [illegible] and thus the overall responsibility is still left to the controllers.

Without changing this fundamental design priniciple of computer assistance for Air Traffic Control, the use of knowledge-based systems instead of pure algorithmic systems now offer new oppurtunities for [illegible] friendly progress of automation in ATC.

Thus it is possible to make use of the knowledge of many different ATCO-experts, in particular of [illegible] operational strategies, heuristics and experience, which are incorporated in a knowle[illegible] [illegible], that allows symbolic processing.

[illegible] benefit of a knowledge based system (KBS) concept is, that it can be easily modif[illegible] or [illegible]

[illegible] the help of the explanatory functions of a KBS, the controller is able to ask for the reasons, [illegible] solution. Thus more transparency and confidence in the computer generated plan can

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

3. Knowledge Based Expert Systems

3.1 General Systems Architecture

Computer-systems that assist, support or partly automatise human activities are [illegible] "expert-systems". An expert systems in general is formed of the following system elements [illegible] knowledge-base, that contains all necessary knowledge, an inference-mechanism, which guides the problem solving process and makes use of the knowledge base. Furthermore an expert system contains a dialogue-element, intended to explain the problem solving process and its results to the human user and - last not least - a knowledge-acquisition-element, that gives the capability to modify or extend the knowledge base.

3.2 Capabilities

The knowledge base may contain a variety of knowledge: textbook knowledge, well defined mathematical functions up to unstructured human heuristics, acquired by experience over years.

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible] taken by [illegible] traffic [illegible] taken [illegible] time [illegible] terminal area, [illegible]

[illegible] area [illegible] Frankfurt Airport [illegible] typical [illegible] from the [illegible] converging at three [illegible]

[illegible] three [illegible] are merged [illegible] the extended runway [illegible] established [illegible] least [illegible] NM [illegible]

[illegible] several [illegible] units, [illegible] of them [illegible] Arrival [illegible] and [illegible]

[illegible]

- [illegible] distributed to several control units.
- [illegible] "stepwise" from the "outer" units to the "inner" units.
- [illegible] local planning prevails in each control unit and must be coordinated [illegible]
- The [illegible] of one overall planning criterion and concerted control action is very difficult to achieve, because of the very high coordination effort.
- The integration of a variety of data from many different sources has to be performed mainly in the brains of the human controllers.

This leads to extremely high work load and even small disturbations which can't be matched, may result in a traffic congestion. Splitting-up and distributing this task to more control units would require even more coordination effort. Therefore it was envisaged to transfer at least parts of the human planning and control functions to a computer.

4.2 Computer-based arrival planning

Based upon studies at Frankfurt Airport, a concept for a computer based planning system (COMPAS), aiming to assist controllers in the comprehensive planning of arriving aircraft was developed, tested and evaluated by the DFVLR Institute for Flight Guidance in cooperation with BFS - the German ATC-Authority /11/.

The essential design principles of the COMPAS-system can be described as follows:

o The stepwise distributed planning of the controllers is substituted by one, comprehensive, overall computer planning.

o The computer planning function anticipates the traffic development for the next 30 minutes and uses one single criterion, common for all units.

o Besides the "usual" data such as radar position information and flight plan data, many other data are included in the computer planning functions, (e.g. traffic load in sub-sectors, aircraft performance and economy, actual airspace-structure etc.). The computer integrates these data and generates concentrated planning results.

o Each control unit involved is provided with its specific planning results, necessary to carry out control and to play its part in the overall plan.

o The controllers stay fully in the loop and keep their executive function. In general the computer generated plan is acceptable to the controllers. However, it is possible for the controllers to interact with the computer in order to modify the plan.

The basic structure of the COMPAS-system is shown in Fig. 3. The operational objectives of the COMPAS-system are with regard to the Frankfurt situation:

o best usage of runway landing capacity,

o delay reduction for arrivals,

o to apply economic descent profiles, if possible.

Fig. 4 shows how the COMPAS-system will be integrated into the existing ATC-system. COMPAS is designed to work in the present ATC-environment. But beyond that, actual radar-data and flight-plan-data are fed on-line into the COMPAS-DP-system via special interfaces. Taking into consideration many other information (aircraft performance, airspace structure, wind etc.) COMPAS generates a plan and displays it to the controllers. The controller may use these COMPAS-proposals, but is not obliged to use the system. However, if he does work with it, the results should be so reasonable and convincing, that he easily can adopt these proposals for his control actions. Under normal conditions no controller-computer interaction is required. However interaction is possible, if the controller wants to modify the plan or if it is necessary to cope with unforeseen events.

The COMPAS-system is still algorithmic - using a branch & bound algorithm - to find the optimum sequence and schedule.

Now as a next step, a rule-based system - called PLANAIR was developed. It is based upon the COMPAS system functions, but instead of using an algorithmic optimization function to determine the optimum sequence, a rule-based production system establishes the proper sequence and schedule, by making use of typical controller strategies and tactics, with regard to the actual traffic situation.

5. The COMPAS/PLANAIR - Systems-Concept

In cooperation between DFVLR-Institute of Flight Guidance and the University of Saarbruecken a rule-based planning system, called COMPAS/PLANAIR was developed (References /9/;/10/).

The rule-based arrival planning system PLANAIR requires certain components of the conventional COMPAS-system. These are in particular all kind of functional/mathematical calculations, such as:

o radar data tracking
o flight plan data procurement
o airspeed calculation from radar data
o economic descent profile calculations
o arrival time prediction.

The results of these calculations are continuously stored in a data-file (FZD in Fig. 5). PLANAIR makes use of this updated data file.

The following description of the PLANAIR component is based upon the work of LUX/9/ and HERINGER/10/.

5.1 PLANAIR-Systems Architecture and Elements

The PLANAIR-systems comprises the typical elements of an expert system, as shown in Fig. 5.:
o an inference element (interpreter),
o a knowledge base,
o a dialogue element,
o an explanation element.

The PLANAIR-system is implemented on a VAX in the OPS5-language. The system comprises about 60 productions and about 30 LISP and C-functions. In order to achieve a more efficient implementation of OPS5, a supplement to OPS5, -the "extended productions"-, were included in the system. They allow LISP-predicates in the condition-part of a production.

In the following chapters the basic elements of PLANAIR are described in more detail.

5.2 The Inference Element

For the dynamic planning with PLANAIR, the OPS5 language was selected, because OPS5 offered the most powerful interpreter and is not limited to specific application areas. Hybrid tools, such as KEE, LOOPS, BABYLON, ART had been under consideration, but have been ruled-out for this prototype system, because of their great complexity. It should be noted, however, that these hybrid expert system shells have a great potential for future applications.

In this paper the OPS5-shell itself shall not be further described. However some of the advantages of OPS5 with regard to the specific requirements of the arrival planning system shall be mentioned.

In the OPS5 productions different instances of various object-classes can be addressed and the correct relation between these elements can be verified.

Apart from making use of logic attributes, OPS5 allows also the processing of objects with numerical values.

OPS5 uses a forward chaining inference mechanism, which is adequate to solve the dynamic planning problem for arrival planning in ATC.

With the so-called "extended productions" in OPS5, it is possible to formulate and include complex procedures and calculations as LISP-predicates as left-hand part of a production rule, thus allowing to include numerical calculations and algorithmic procedures (e.g. profile calculations), which are an integral part in air traffic planning. They can be processed in the normal rule matching process of the OPS5-interpreter.

Some of the disadvantages of OPS5 should also be mentioned. A main handicap for OPS5 is the lack of a user friendly programming environment and the lack of comprehensive debugging support.

An essential shortcoming of the OPS5 interpreter for dynamic planning is, that the rule-matching process can't be interrupted, but has to be completed in any case! This means in the case of ATC-planning, that relevant new information (e.g. new aircraft; new position; controller inputs) can't be considered in the running cycle, but only after its completion, in the next cycle.

5.3 The Knowledge Base

The PLANAIR knowledge base is formed of three parts:

o declaritive knowledge, describing the air traffic environment,

o procedural knowledge, i.e. the planning rules,

o text elements for the explanation functions.

5.3.1 Air Traffic Knowledge

The knowledge about the air traffic is represented in the knowledge base by working memory elements as part of the OPS5-production system.

Static knowledge, that does not change, is formed by information on the airspace structure, aircraft performance, separation data etc. The declaration in the working memory is made by the "literalize" statement, by which a class (i.e the class "aircraft") is specified with all its necessary attributes.

Dynamic knowledge on the air traffic, which changes continuously, such as: position, speed, altitude is stored in the same way as working memory elements. However, this part of the working memory is continuously updated.

5.3.2 Production Rules - The Procedural Knowledge for Arrival Planning

The knowledge that is necessary to solve the arrival planning problem, is coded in OPS5 production rules.

In a process of knowledge acquisition with the help of ATC-staff from the Frankfurt Air Traffic Control Center some basic human controller strategies were identified and formulated as "rules", i.e. "productions" in the OPS5 language.

It should be noted that these rules are not yet comprehensive. Additional knowledge still has to be acquired. However the rules found so far, already form a solid base to establish an overall arrival sequence and schedule, that is efficient with respect to traffic demands and reasonable with respect to human controllers reasoning and decision making.

In the knowledge acquisition process it became evident, that the various controller strategies or "rules of thumb" can be classified in four groups, i.e. rules, that refer to:

- o the overall planning strategy,
- o tentative sequence planning,
- o tentative arrival time planning,
- o final sequence/schedule planning.

These four groups of planning rules form a four level processing-hierarchy. This means, that in addition to the basic rules gained from the controllers, some more rules (meta-rules) for the overall control hierarchy had to be set up, which allow to define, to modify and to consider the hierarchy level of each strategy. The productions are processed according to their hierarchy levels. This concept allows to assign a higher priority to specific strategies, depending on the traffic situations, or to adapt to individual human controller preferences.

5.3.2.1 Overall Planning Strategy

PLANAIR presently provides two main planning strategies. One making use of every means to expedite the flow of arrivals, i.e. the planning process is based on the "Earliest Estimated Arrival Times". This strategy may be used in peak-periods.

The other strategy uses the "Estimated Arrival Times", based upon economic idle descent profiles. Fig. 6 gives an impression how these two strategies are expressed in an OPS5 production.

5.3.2.2 Sequence planning strategies

To establish a proper sequence of arriving aircraft 9 different sequencing strategies have been implemented in PLANAIR so far. According to the planning hierarchy concept each strategy has to be assigned a priority level.

The nine sequencing strategies are as follows:

- o The "First-Come-First-Served"- Strategy, which is the basic strategy. All aircraft are intially planned according to this strategy, before other sequencing strategies may be applied.
- o The "Delay"-Strategy
  Using the delay strategy, PLANAIR gives preference to the aircraft with the greatest delay versus its scheduled arrival time.
- o The "Short-Cut"-Strategy
  In low density periods controller tend to apply path shortening manoeuvers. An aircraft is planned to use an assigned short-cut; the potential time-saving is stored in the air traffic knowledge base.
- o The "String"-Strategy
  Controllers usually try to establish an in-trail sequence of aircraft with proper longitudinal separation. The control effort for such "string" then is lower.
  The PLANAIR "string-strategy" tries to maintain strings of aircraft, if no other aircraft or strategies are penalized or violated.
- o The "Pile"-Strategy
  Aircraft merging on a waypoint at the same time, however vertically separated, are often handed over to approach control as a "pile". The approach controller then gives different vectors in order to establish the necessary separation. If possible, he tries not to merge other traffic into this group of aircraft. PLANAIR provides the same strategy option.
- o The "Low-Before-High" Strategy
  For several reasons a controller usually establishes a sequence with the lower aircraft first, followed by the higher aircraft. This strategy is part of the rule-base.
- o The "Height-Rule"-Exception
  This rule allows to abrogate the normal "Low-Before-High"-Strategy, e.g. if the low aircraft is very slow, and a conflict-free descent profile for the faster but higher aircraft is possible.
- o The "Sect -Priority"-Strategy
  Due to unbalanced traffic streams from different directions, traffic demand in a certain sector can be much higher then in others. In order to avoid congestion, aircraft from high density sectors are prefered.

Each of these strategies is stored in the knowledge base in the form, of one or more OPS5 productions. An example for the "Low-Before-High"-Strategy is given in Fig. 7.

5.3.2.3 Arrival Time Planning Rules

After the sequence planning the working memory elements for alle known arriving aircraft are ordered according to their assigned sequence time. The time planning function then assigns the planned arrival times for each aircraft, taking into account the necessary separation to the predecessor, as well as the possible delay. If a time-conflict with the succeeding aircraft is created, all time-conflicts for the succeeding aircraft then have to be checked and to be resolved.

5.4 The Planning Cycle

The OPS5 interpreter runs stepwise through all productions in the four described hierarchy levels. The inference mechanism initially tries to find out, whether the condition-side of a production is matched. All productions that "match", form the aggregate of productions which may potentially satisfy the conditions.

All these possibly conflicting productions are then handed over to a conflict resolution mechanism, which selects the applicable rule according to specific criteria or stops when there are no more [illegible] left. The interpreter cycle finishes with the "action-phase", i.e. the right hand side [illegible] of the production rules are carried out, e.g. working memory elements are modified.

5.5 The Dialogue Element

For the display of results and the dialogue with the controller a special [illegible] interface was developed for a TI/EXPLORER computer.

The display is divided up into several windows (Fig. 8)

o Window for a radar-like Air Traffic Display (upper left),
o Window for the PLANAIR-generated Time Schedule and Sequence (upper right),
o Window for Trace of OPS5,
o Window for interaction menue etc. (bottom).

The "interact"-menue offers the following options to the controller

o Explanation (explanation about the sequence),
o Insert Slot (for additional aircraft, i.e. missed approaches),
o Delete Slot (for cancelled flights),
o Sequence Change (if he does not agree with the PLANAIR proposal),
o Runway Change (at any time),
o Flow Change (airport acceptance/landing rate).

5.6 The Explanation Component

The explanation Component of the knowledge base contains text-elements for the explanation of PLANAIR reasoning and planning results:

A text element has the following structure.

o class of text element
o conditions for text element
o string of text with placing parameters
o placing parameter 1
o placing parameter 2
o placing parameter 3.

For each of the last 10 planning cycles the planning information about all active aircraft is stored in a list-file. A list element for one cycle contains the following information.

o Individual Code
o Planning Status
o Standard Arrival Route (STAR)
o Earliest Estimated Arrival Time
o Estimated Arrival Time
o Planned (Assigned) Arrival Time
o Priority level/class of text element.

With this information and the conditions of the text elements, all explanatory text can be generated if the controller asks for an explanation. The controller calls-up "explanation" and "clicks" the Individual Code(s) of the aircraft. Then the list-file elements with the desired individual codes are called for the corresponding planning cycles. The planning information on that aircraft, which is stored in the list-file, is then matched against the class- and the condition-parts of the text elements. According to the productions that matched and the placing parameters the explanatory text in selected and displayed.

[illegible] Conclusions

With the described PLANAIR-System a first step was made to build a rule-based system, intended [illegible] assist air traffic controllers in their arrival sequencing and scheduling planning function.

Contrary to planning systems, where the sequence is established by an algorithm using mathematical models and functions, the PLANAIR-system establishes a proper sequence, according to rules (strategies, tactics, experience), which are used by air traffic controllers.

The experience and results gathered so far can be [illegible] as follows:

A proper, safely separated sequence can be established with PLANAIR.

In comparison with the algorithmic COMPAS-system, PLANAIR establishes a different sequence [illegible] COMPAS tries to minimize the total delay, while PLANAIR presently does not [illegible] optimization [illegible]. Instead of it, PLANAIR suggests a plan, that corresponds largely to the procedures usually used by controllers. It can be expected that the PLANAIR-generated plan is more acceptable for the controller and needs less control effort.

[illegible] PLANAIR, [illegible]-based expert system was developed, which is able to continuously perform [illegible] knowledge base and data base that changes dynamically over time. As mentioned [illegible] that a planning cycle [illegible] be interrupted, but has to be completed, even if new data [illegible] which might have [illegible] for the plan.

Presently, the computation time for the [illegible] PLANAIR [illegible] much higher than the computing time for the [illegible] FORTRAN program for COMPAS [illegible].

[illegible] system, thus the effort to test and implement [illegible] complex [illegible].

References

[illegible] Völckers, [illegible] "Computer Assisted Arrival Sequencing with the COMPAS-System". AGARD [illegible], 1986, [illegible].

[illegible], A.; [illegible] "Next Generation of Control Techniques in Advanced TMA". AGARD [illegible] 1986, pp. [illegible].

[illegible] "Planning in the world of the Air Traffic Controller". International Joint Conference on Artificial Intelligence, [illegible], pp. [illegible].

[illegible] "Problem Solving with Simulation in the world of an Air Traffic Controller". PhD Dissertation, University of Texas, Austin, [illegible].

[illegible] "Qualitative Reasoning in an Expert System Framework". PhD Thesis, University of Illinois, Urbana, 198[illegible].

[illegible] "An Examination of Distributed Planning in the world of Air Traffic Control". Arizona State University, AI-Lab., 1984.

7. Tobias, [illegible]; Scoggins, [illegible] "Time-Based Air Traffic Management Using Expert Systems". NASA-TM-88234, 1986.

8. [illegible]llis, A. [illegible] "Rule-Based Implementation of Automation Aids for ATC-Controllers". FTL-Report R85-[illegible], Flight Transportion Lab., MIT; Cambridge, 1985.

9. [illegible]ax, A. "PLANAIR - Vergleich eines algorithmisch-technischen Ansatzes zur Anflugplanung mit einem wissensbasierten Planungsansatz". Diplomarbeit under supervision of Prof.W. Wahlster, University of Saarbruecken, Germany, 1987.

10. Peringer, G. "Wissensbasierte Anflugplanung fur Verkehrsflughafen - am Beispiel des Flughafens Frankfurt/Main". Diplomarbeit under supervision of Prof.W. Wahlster, University of Saarbruecken, Germany, 1987.

11. Volckers, U.; Schubert, M.; Schick, F.V.: "Ergebnisse der Simulatorerprobung des Planungssystems COMPAS". DFVLR IB-112-87/02, Braunschweig, 1987.

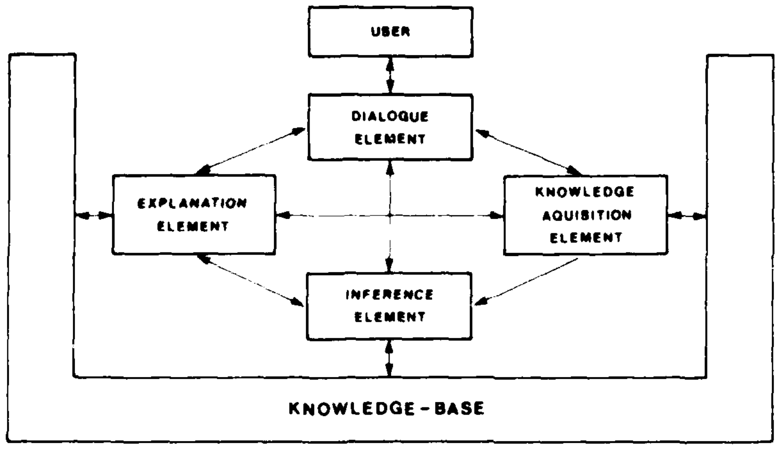


Fig. 1 Expert System Elements

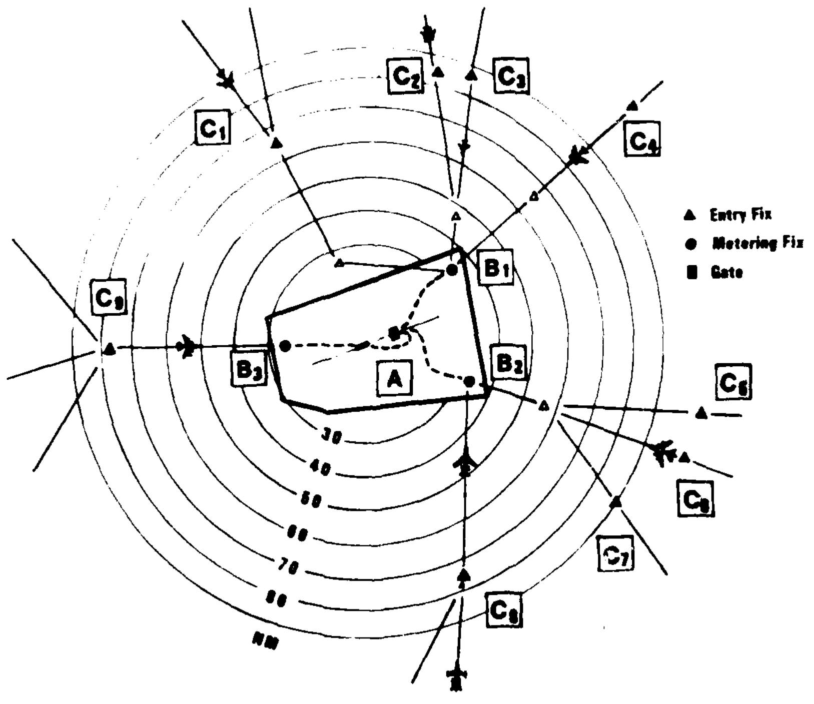


Fig. 2 Airspace Structure of Frankfurt Approach Area

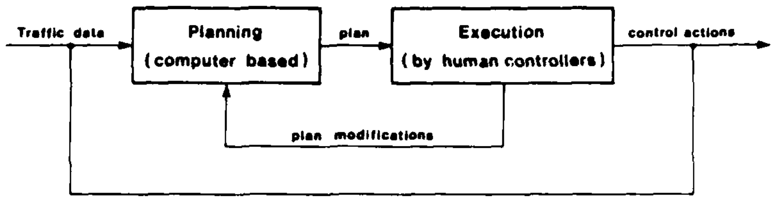


Fig. 3 Planniı g and Execution Functions with COMPAS

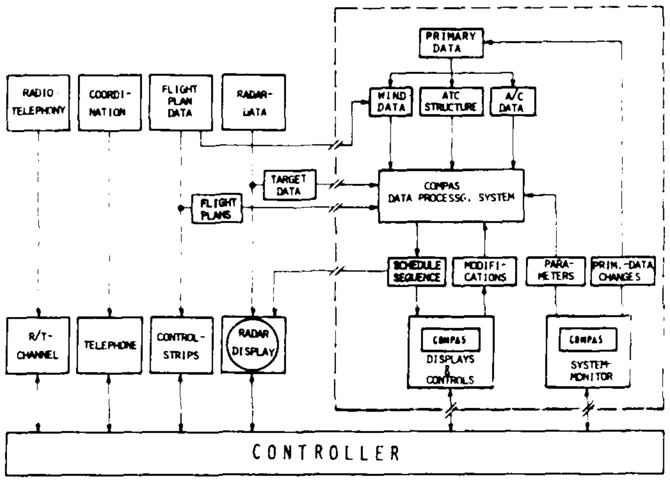


Fig. 4 COMPAS - Computer Assistance in ATC

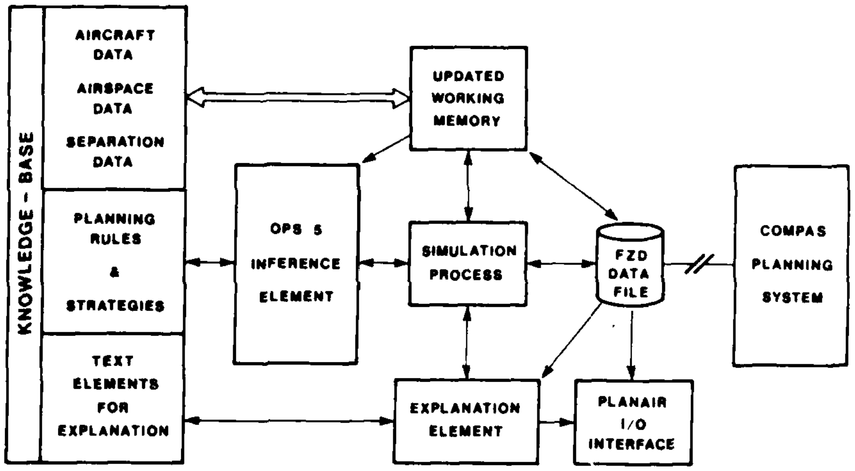


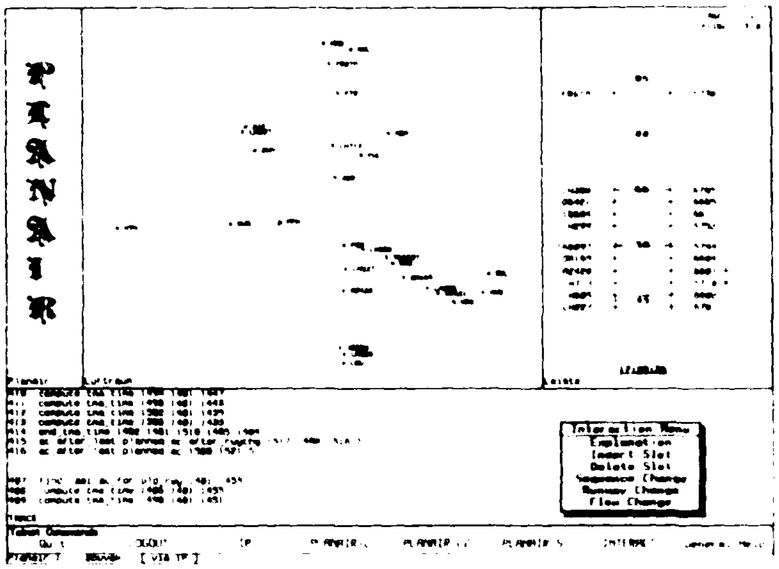
Fig. 5 COMPAS PLANAIR Systems Concept

```
(p switch_to_seq_plan_with_acc
   (prod ^ prod_name switch_speed ^ prod_pri <pri>)
   (control_flag ^ c_value {<pri> >= 0}
                 ^ c_step <step>
                 ^ c_acc without_acc)
   (aircraft ^ ac_delay {> 66}
                 ^ ac_status {<> 4}
                 ^ ac_acc_mode {<> x}
                 ^ ac_seq_time_gt <stgt1>)
   (aircraft ^ ac_seq_time_gt {<stgt2> <= <stgt1>}
                 ^ ac_plan_time_gt <ptgt2>
                 ^ ac_status {<> 4}
                 ^ ac_acc_mode {<> x}
                 ^ ac_delay {<= 0})
   - (aircraft ^ ac_seq_time_gt {> <stgt2> < <stgt1>}
                 ^ ac_delay {<= 0})
   - (aircraft ^ ac_plan_time_gt {< <ptgt2>})

-->

   (modify 2 ^ c_value remove_seq_time_gt
                 ^ c_acc with_acc)
   (modify 3 ^ ac_seq_time_gt nil ^ ac_plan_time_gt nil
                 ^ ac_delay nil)
   (modify 4 ^ ac_seq_time_gt nil ^ ac_plan_time_gt nil
                 ^ ac_delay nil)
   (make proof_sequence ^ pr_pre <stgt2> ^ pr_suc <stgt1>))
```

Fig. 6 OPS5 - Productions for Overall Planning Strategy

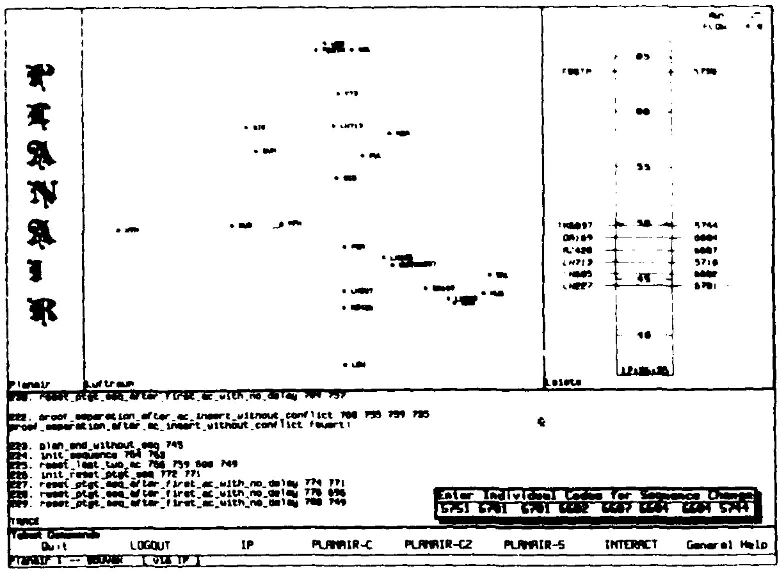
```
(ep two_ac_in_same_sec_with_same_time_deep_before_high_cond
    (prod ^ prod_name height_rule ^ _prod_pri <val>)
    (control_flag ^ c_value {<val> >= 0} ^ _c_step <step>)
    (aircraft ^ ac_seq_time_gt {<stgt1> <> nil}
              ^ ac_status {<> 4}
              ^ ac_star <star1>
              ^ ac_last_height <height1>
              ^ ac_prio {<= <val>}
              ^ ac_plan_time_gt {<> new})
    (aircraft ^ ac_seq_time_gt {<stgt2> <= <stgt1>}
              ^ ac_status {<> 4}
              ^ ac_star <star2>
              ^ ac_last_height {> <height1>}
              ^ ac_prio {<= <val>}
              ^ ac_plan_time_gt {<> new})
    (star ^ star_nr <star1> ^ _st_mf_name <mf1>)
    (star ^ star_nr <star2> ^ _st_mf_name <mf1>)
    ($eqtime_p$ 3 ac_seq_time_gt ac_seq_time_gt
              (call identity <stgt2> <stgt2>))

-->

    (make proof_sequence ^ pr_pre <stgt2> ^ _pr_suc <stgt1>)
    (modify 3 ^ ac_plan_time_gt new ^ _ac_prio <val>)
    (modify 4 ^ ac_seq_time_gt (compute <stgt1> + 1)
              ^ ac_plan_time_gt new ^ _ac_prio <val>)
    (modify 5 ^ ac_seq_time_gt (compute <stgt2> + 2)
              ^ ac_plan_time_gt new ^ _ac_prio <val>))
```

Fig. 7 OPS5 - Productions for "Low-before-High"-Strategy



```
[illegible] compute tma_time [illegible]
[illegible] compute tma_time 1490 1481 1448
[illegible] compute tma_time 1500 1481 1439
[illegible] compute tma_time 1500 1481 1480
[illegible] end_tma_time 1492 1481 1510 1485 [illegible]
[illegible] ac after last planned ac after rwychg [illegible]
[illegible] ac after last planned ac 1500 [illegible]

[illegible] find last ac for pld rwy 1481 1459
[illegible] compute tma time 1490 1481 1459
[illegible] compute tma_time 1490 1481 1451
TRACE
```

Fig. 8   PLANAIR-Display with Interaction Menue



```
220. reset_ptgt_seq_after_first_ac_with_no_delay 794 757

222. proof_separation_after_ac_insert_without_conflict 700 755 759 795
proof_separation_after_ac_insert_without_conflict fewert!

223. plan_end_without_seq 745
224. init_sequence 764 763
225. reset_last_two_ac 766 759 800 749
226. init_reset_ptgt_seq 772 771
227. reset_ptgt_seq_after_first_ac_with_no_delay 774 771
228. reset_ptgt_seq_after_first_ac_with_no_delay 776 896
229. reset_ptgt_seq_after_first_ac_with_no_delay 780 749
TRACE
```

Enter Individual Codes for Sequence Change
5751 6701 6701 6682 6687 6604 6604 5744

Fig. 9   PLANAIR-Display (Sequence Change)

[illegible]

PROLOG : An Implementation Language

[illegible] expert system.

Extensions to PROLOG

[illegible]

As a result, since we were faced with the problem of developing expert systems in an operational context, we developed the [illegible] environment [illegible] based [illegible] PROLOG as its implementation language. Many facilities were introduced in this environment which made the writing of meta-rules easier, and allowed the successful development of several expert systems.

However, as soon as we had to go beyond the basic formalism (to implement a specific search strategy, for example), we had to resort to either the formalism ([illegible]) or to the implementation language (PROLOG), which sometimes was difficult to do or proved ineffective. That's why we decided to define a new environment, [illegible] ([illegible]), in which we tried to stress facilities required to develop expert systems in an industrial context, namely :

. possibility of implementing a wide range of knowledge representation formalisms,
. mechanisms for defining strategies specific to the problems to be dealt with,
. fast and cheap development of expert system prototypes.

[illegible]

[illegible] towards an object-oriented representation on which [illegible]. This was motivated by the fact that such a represen- [illegible] (factual knowledge, constraint propagation...) but also [illegible] insertion and deletion of PROLOG clauses.

[illegible] imagine representing the fact base [illegible] passed as an argument and recursively [illegible] acceptable memory requirements). Hence it [illegible] asserted or retracted during reasoning. Now, when a [illegible] with respect to the resolution [illegible] behaviour during backtracking on a deleted clau- [illegible] This is why it seems to us that the representation

[illegible] deletion of a clause,

[illegible] as well as changing the knowledge base,

[illegible]

[illegible] has been based on a PROLOG extension towards an object [illegible]. This extension, principal characteristics of which [illegible] designed for the development of expert systems. As a result, [illegible] facilities, it has some elementary mechanisms that are [illegible] strategies specific to the problems (intelligent backtracking, [illegible] be difficult to implement directly in PROLOG.

[illegible] might be possible to program an expert system directly in this new [illegible] "the environment for production rules" built around the [illegible], which is described in-depth in Chapter 3. This environment [illegible] the development of expert systems (essentially a framework for the descrip- [illegible] to mask the underlying language.

[illegible] : OBJECT ORIENTED REPRESENTATION

[illegible]

[illegible] representation is based primarily on the concept of "frame" [MINSKY 75] (and, in particular, [illegible]) which is both a powerful and effective concept for the representation of fac- [illegible] low-level reasoning in expert systems [FARGUES 83].

[illegible] existing in the FRL language are removed from EMICAT (MI4). For example, the [illegible] relationship is no longer defined by a special attribute (ISA) but by an explicit rela- [illegible], thereby avoiding ambiguities which were cropping up during the "inheritance" process.

Moreover, procedural attachment capabilities were extended so that a more precise specification of the moment of activation could be provided. As a result, the following procedures are attachable to any at- tribute :

. **if-needed,**

. **before-insertion,**

. **after-insertion,**

. **before-deletion,**

. **after-deletion,**

. **before-change,**

. **after-change.**

Finally, some new facilities were introduced, among which :

- the inheritance scope specification (to enhance efficiency),

- an anti-looping checking at the procedural attachments level, [illegible] defined backward-chaining strategy easier - (refer to Chapter [illegible]),

- a state memorization mechanism (described in the next [illegible]

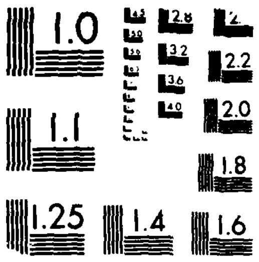
AD-A191 463 KNOWLEDGE BASED CONCEPTS AND ARTIFICIAL INTELLIGENCE: APPLICATIONS TO GUIDANCE AND CONTROL(U) ADVISORY GROUP FOR AEROSPACE RESEARCH AND DEVELOPMENT NEUILLY... 2/2

UNCLASSIFIED AUG 87 AGARD-LS-155 F/G 1/4 NL

END

DATE FILMED

88



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963 A

2.2 STATE MEMORIZATION MECHANISM

An expert system development tool must offer the following three choice mechanisms [LAURENT 84] :

- choice of the state of the knowledge base on which to act,

- choice of the rule to be applied,

- choice of the objects on which to apply the rule.

The first of these choices is the most hard to implement and costly to use because it requires often bulky information storage ; that's why few development tools propose it.

It happens that if this mechanism is integrated with the object representation, the expert system designer has an elementary tool that allows him to program strategies tailored to the problem to be treated. To do so, in EMICAT (MI4), a prototype called "state" is predefined.

Operations carried out on objects are constantly memorized inside the instance of prototype "state" labelled as "current-state". Every time a backtrack point has to be created, a new instance of "current-state" must be generated. Such a mechanism makes it possible to memorize the search tree so as to be able to backtrack to any state.

Two primitives suffice to control this mechanism :

- new-state(x), which instanciates a new object, from the "current state",

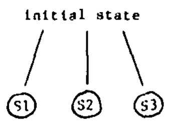
- goto(x), which allows changing of current state.

Example :

In order to implement a strategy for selecting the best rule from a list L, when only the evaluation of the rule allows judging its interest, it is sufficient to write (upper-cases designating the PROLOG variables) :

```
strategy (L)<--- member (R,L)  &     successively selects the rules to be tried

                 new_state (S) &     creates a new state ; return to the previous state during
                                     backtrack

                 apply (R)     &     applies the rule

                 evaluate (S)  &     evaluates the current state and store the results of the
                                     evaluation in an attribute of the object S

                 fail                backtracking

strategy (L)<--- select (S)    &     selects the best state by analyzing the stored attributes

                 go_to (S)           fixing of the selected state
```

The generated state tree is in this particular case :



The information stored in each object and enabling a state change is user-accessible and allows programming a strategy such that, for example, in the event of a deadlock condition the system returns to the state having generated a given attribute as indicated in [VIALATTE 85].

Since an expert system may not necessarily resort to such a memorization in all the reasoning phases, this mechanism may be turned off so as to not penalize all the searches.

CHAPTER 3 : ENVIRONMENT FOR PRODUCTION RULES

The object oriented representation alone is not enough to develop an expert system. In fact, a large amount of knowledge is represented more easily in the form of production rules. This is why EMICAT (MI4) which is built around the "PROLOG-object oriented representation" pair, includes an environment that allows defining and using production rules.

3.1 RULES WITH EMICAT (MI4)

The definition of a production rule formalism is subject to a number of often antagonistic constraints, that is :

. the formalism must be powerful and general, but it must also be adaptable to a particular expert field so that the expert himself can enter the rules.

. the formalism must allow rules to be written but also meta-rules, the former dealing mostly with the expert field and the latter with the search strategy.

. the formalism must be declarative, but it must also allow expressing knowledge of a procedural type (search strategies).

To cope with these constraints, the basic form of an EMICAT (MI4) rule is a conjunction of PROLOG goals.

For example, suppose the rule is expressed in English as follows :

"If the airplane searched for is a delta wing and carries passengers, concorde should be added to the list of suspects",

then this rule may be written as :

```
Value     (airplane_searched_for, wing, delta) &

Value     (airplane_searched_for, passenger, yes) &

Value     (suspect, list, L) &

Set-val   (suspect, list, concorde.L).
```

It is clear that such a formalism although convenient for the knowledge engineer who has all the power of PROLOG at his disposal does not make exchanges with the expert easy and, in any case, does not allow the expert to enter the rules himself. This is why the previous form represents only the internal form of the rules, usable for resolving certain specific types of problems - such as programming a special search strategy ; in practice, the user can define each rule as an object, inheriting from the general "rule" prototype and having two attributes :

- one describing the "body" of the rule, provided in a form best adapted to the problem,

- the other describing the "translation" of the rule, applied to the body to convert it into a PROLOG goal conjunction (internal form).

With this approach the hierarchical relationship between objects may be used to define families of rules the formalism of which is adapted to the type of knowledge to be represented ; often the same expert system should resort to several formalism simultaneously.

The translation is automatically carried out when a rule is entered or changed by the set of procedural attachments inherited from the highest level prototype ; thanks to the use of operators present in most PROLOG implementations, this translation may remain simple, while providing the expert with an easy-to-manipulate formalism. For example, the rule used in the previous example will be written as :

```
object rl

  body if wing = delta and passenger = yes then suspect (concorde)

       translation tr (External-rule, Internal-rule)

  with simply for the translator :

  tr(if P then suspect (X), Pl &
       value (suspect, list, L) &
       set-val (suspect, X.L)
                     trl (P, Pl).

trl(A=B and R, value (airplane_searched_for,A,B) & Rl)   /  & tl  (R, Rl).
trl(A=B, value (airplane_searched_for,A,B)).
```

REMARK :

It might be surprising to find no distinction between the condition and action parts of rules, which are the very essence of production rules ; in particular, it might seem problematic for determining the conflict-set formed by rules applicable at any given time.

In practice, one can always re-establish this distinction by defining an external ad-hoc format ; moreover, the many possibilities allowing the manipulation of rules in EMICAT (MI4) and described in the next paragraph, constitute a powerful and well-adapted tool to cope with rule selection.

3.2 HOW TO "TALK ABOUT" RULES

The importance of meta-knowledge doesn't have to be proven anymore [PITRAT 82] ; in fact, as soon as an expert system reaches roughly a thousand rules, the application mechanisms, regardless of the associative addressing set up, remain essentially sequential and loose their efficiency.

To aid in determining what knowledge is useful for resolving a given problem, EMICAT (MI4) offers five possible ways of referring to the rules present in the base :

1) organize the rule in a hierarchical form, which is achieved immediately thanks to the object representation on which the rules are constructed.

2) construct one or more index(es) ; the construction is automatically carried out at the moment when the rule is entered according to the instructions of an attribute specifying :

   . the index name,

   . the pattern to be used as a key,

   . the pattern search mode (total unification, filtering in the direction rule-pattern or in the direction pattern-rule).

3) refer to a set of attributes computed when the rule is entered and marking the constants that are specified therein (objects, types, attributes, external predicate) allowing meta-rules comparable to [DAVIS 80].

4) refer to attributes entered by the rule's writer (to specify, for example, the cost or the importance of the rule).

5) unification or filtering on the rule's body (external form).

   Using the last way, any information present in the rule is accessible. However, for the sake of efficiency and effectiveness it is clear that it should preferably be used in conjunction with the other ways so that a subset of pertinent rules can be rapidly selected.

3.3 APPLICATION OF RULES

The main objective of EMICAT (MI4) is to allow defining strategies adapted to the problem to be solved ; this is made possible by a unique rule-application primitive described in the next paragraph, which is used for predefined strategies (§ 3.3.2) as well as user strategies (§ 3.3.3).

3.3.1 Application Primitive

The application primitive is a predicate with three arguments and is written as follows :

apply (rule-choice, object-choice, selected-object)

Each solution corresponds to a successful instanciation of one of the selected rules. The different solutions are obtained - which is always the case in PROLOG -whenever backtracking occurs on the primitive. The three arguments have the following meanings :

- the **rule-choice** is made either by giving the list of rules to be applied or by a property (conjunction of PROLOG goals) that should be verified by them, as for example :

  R:: object (R, diagnosis) & value (R, cost, low)

  allows applying all the rules the type of which is "diagnosis" and the cost of which is "low".

- the **object-choice** argument specifies, at the time of application, on which objects the rule has to be applied. Indeed, it is highly advantageous to be able to state general knowledge in the form of a rule (like Ohm's law), but to apply this knowledge to the only objects pertinent to the problem to be solved (and not as in the example of Ohm's law to all the components of an electronic circuits !).

  Thus, for example :

  Object (R, resistor):: value (R, power_max,X) & X > 2

  will limit the application of the rule to resistors the power_max of which is greater than 2 watts.

- the **selected-object** argument allows communication between the rule and the application mechanism ; to do so, the rule's writer can specify with a predefined predicate (select) a term in the rule that will be unified with the third argument of the application primitive.
  Thus, for example, to "apply rules selected by meta-rules m1 and m2", it will suffice to write :

  apply (R:: apply (m1.m2.nil, nil,R),nil,*)

  provided that meta-rules m1 and m2 encompass a term such a "select('rule to be selected')".

3.3.2 Predefined Strategies

EMICAT (MI4) offers a few predefined strategies built thanks to the primitive "apply", such as :

Pass	: evaluate all the solutions
Saturate	: successively evaluate "pass" until no object is changed any longer during a pass
Prove	: apply until saturation is reached or until a goal (conjunction of PROLOG goals) supplied as an argument is proven.

3.3.3 User Strategies

It is not by chance or omission that we have not discussed backward chaining up to now. First of all, backward chaining *must not be considered* as a mode of application for a rule (since the rule can only be applied rigorously in the direction condition - action) but rather as a search strategy. On the other hand, the experience that we have acquired from the development of several expert systems has demonstrated to us that it was simply not realistic to propose a "standard" backward chaining ; in fact, the number of parameters to take into account is very large :

- should the search be carried out in depth-first or broad-first strategy ? (or a mix of both) ?

- should all the objects be searched for ? (for some it often makes no sense).

- when and for which objects should the operator be queried ?

This is why we think that it is up to the knowledge engineer to define the backward search strategy. It also seems to us that EMICAT (MI4) offers a large amount of facilities to implement such a strategy. Thus, for example, to carry out :

- the proof of a goal of the form : value (X,Y,Z)

- in depth-first strategy

  without user querying

it suffices :

- to provide for the generation of an index (see § 3.2) of the rules mentioning the term set-val (X,Y,Z)

- to attach to the objects for which backward chaining is desired the procedural attachment if-needed with the value :

```
parm (OB, ATT, Val) &

apply (R::index(set-val(Ob,ATT,Val),R),nil,*)
```

and that's it ! The evaluation of a goal of the form : value (X,Y,Z) - either when a search is initiated or when a rule is evaluated - will generate a search by the application of rules that are liable to prove the target when it is not present in the base ; the anti-looping system integrated with the object representation prevents the possibility of a potentially infinite search.

CHAPTER 4 : IMPLEMENTATION

The environment just described was developed on an IBM 30xx with the language VM/PROLOG. The essentials of this environment reside in the object oriented representation described in Chapter 2 ; considerable stress was placed on ensuring good response times. As a result, the value of an attribute is accessed in roughly 100 microseconds. Obviously, if the object representation were totally integrated in the PROLOG interpreter, these response times could be dramatically improved.

And, for the production rules, they are "compiled" or, to be more precise, "converted" so as to introduce as efficiently and effectively as possible the object-choice mechanism described in § 3.3.2. It should be noted that the "compiler" can be redefined by a knowledge engineer when a particular need warrants it.

CONCLUSION

The purpose of the EMICAT (MI4) environment just described is to use the incontestible advantages of the PROLOG language to benefit expert system programming.

The more salient features of EMICAT (MI4) are :

- object-oriented language features
- integration to PROLOG environment
- easiness for programming strategies suited to various applications (including intelligent back-tracking and meta-rules writing facilities)
- flexibility for defining various rule formalisms

But, above all, EMICAT (MI4) is an industrial product tailored to ease the development of knowledge based applications. For that purpose, various facilities are provided, such as :

- integration to graphics
- trace (at various levels)
- object editing
- knowledge base archiving

Since the beginning of 1986, EMICAT (MI4) has been used for more than 10 applications, in four major fields :

- military (battlemanagement, avionics)
- space (satellites : battery management, diagnostic)
- software engineering (quality assurance, rapid prototyping)
- CAD/CAM (layout verification, diagnostic)

Most are expert systems, though some (a fault tree generator for avionics, rapid prototyping) use mainly the object-oriented features of EMICAT (MI4) for knowledge representation.

REFERENCES

[CLARK 82] K.L. CLARK, F.G. Mc CABE, S. GREGORY
I.C. PROLOG Language Features
in LOGIC PROGRAMMING
Academic Press 1982

[CORDIER 83] M.O. CORDIER, M.C. ROUSSET
TANGO : Moteur d'inférences pour un Système Expert avec variables
Rapport de recherche n° 123
LRI ORSAY, 1983

[DAVIS 80] R. DAVIS
Content reference : reasonning about rules
Artificial Intelligence, Vol 15, n°3, pages 223-239
Décembre 1980

[DINCBAS 83] M. DINCBAS
Contribution à l'étude des Systèmes Experts
Thèse de Docteur Ingénieur
Ecole Nationale Supérieure de l'Aéronautique et de l'Espace
Janvier 83

[FARGUES 83] J. FARGUES
Contribution à l'étude du raisonnement ; Application à la médecine d'urgence
Thèse d'Etat Université PARIS VI
1983

[FORGY 81] C. FORGY
The OPS5 user's manual
CMU-CS81-135
Carnegie Mellon University,
1981

[HAMMOND 84]	P. HAMMOND, M. SERGOT APES : Augmented Prolog for Expert Systems Reference Manual, Micro-Prolog Version Logic Based Systems Ltd RICHMOND, SURREY, UK, 1984
[LAURENT 84]	Jean-Pierre LAURENT La structure de contrôle dans les Systèmes Experts TSI, Volume 3, n° 3, pages 161-177 1984
[LAURIERE 83]	J.L. LAURIERE SNARK : Un moteur d'inférences pour Systèmes Experts en logique du premier ordre Rapport 430 Institut de programmation Université de PARIS VI 1983
[MELLE 81]	W. VAN MELLE, A. SCOTT, J. BENNETT, M. PEAIRS The EMYCIN manual Report n° STAN-CS-81-885 Department of Computer Science STANFORD UNIVERSITY, 1981
[MINSKY 75]	M. MINSKY A framework for representing knowledge In P. WINSTON : the psychology of computer vision New-York Mc GRAW-HILL 1975
[OGAWA 84]	Y. OGAWA, K. SHIMA, T. SUGAWANA, S. TAKAGI Knowledge representation and inference environment : KRINE an approach to integration of frame, Prolog and graphics International conference on fifth generation computer systems TOKYO, Novembre 1984
[PITRAT 82]	J. PITRAT Les connaissances déclaratives Colloque Intelligence Artificielle Publication n° 30 du GR 22 - CNRS LE MANS, Septembre 1982
[ROBERTS 77]	R.B. ROBERS, I.P. GOLDSTEIN The "FRL" Primer The "FRL" Manual Memo 408 et 409 Massachussets Institute of Technology 1977
[TAILLIBERT 85]	P. TAILLIBERT MI3 - Outils de représentation et d'exploitation de la connaissance 5ème Journées Internationales sur les Systèmes Experts et leurs applications AVIGNON (FRANCE), Mai 1985
[TAILLIBERT 86]	P. TAILLIBERT, S. VARENNES MI4 ou comment développer des systèmes experts avec PROLOG 6ème Journées Internationales sur les Systèmes Experts et leurs applications AVIGNON (FRANCE) Avril 1986
[VIGNARD 85]	P. VIGNARD Un mécanisme d'exploitation à base de filtrage flou pour une représentation des connaissances centrée objet Thèse de Doctorat Institut Naitonal Polytechnique de Grenoble Juin 1985
[VIALATTE 85]	M. VIALATTE Description et Application du moteur d'inférence SNARK Thèse de Doctorat Université PARIS VI Mai 85

REAL-TIME EXPERT SYSTEMS: A STATUS REPORT

B. A. Bowen, Ph.D., P. Eng.,
Professor, Department of Systems and Computer Engineering
Carleton Universtiy, Ottawa Canada, K1S 5B6

SUMMARY

The control algorithm for complex physical plants is usually based on a model, either discrete or linear, which describes the appropriate response to data obtained from sensors located throughout the plant. Models of plant behaviour are usually based on a Markov process assumption or on some linear system approximation. Many subtle characteristics beyond the range of the assumptions or approximations are either ignored or the appropriate response is obtained by fine-tuning after the control system is installed.

Plant representation and control by a knowledge-based system is an attractive alternative. A knowledge base can reflect not only the algorithmic aspects of a plant's behaviour, but take into account many factors which are difficult or impossible to describe algorithmically. In addition, such systems allow strategic considerations to be incorporated in the controller, which would be impossible to realize in any other fashion. Such systems are being implemented in growing numbers.

This paper presents a comprehensive state-of-the art summary of the status, progress, issues and directions in the use of knowledge-based systems in real-time control applications. It includes a bibliography of current work, up to January 1987.

0.0 A Perspective on Control Systems

The description of a physical plant for the purposes of computer control is often implemented as a Markov process represented by a finite-state machine or by an equivalent algorithmic representation of the control states and the transition requirements. In a purely sequential control situation, the design is relatively simple in that the correct procedure is instituted according to a preset time standard or in response to anticipated events during operation. As the topology of the control algorithm becomes more complex, the representation and execution constraints become increasingly difficult to implement.

In the situation where many events can occur asynchronously and where each must be attended to in some fixed period of time, the controller design becomes even more constrained. The designer must guarantee adequate responses within the specified time interval for the worst case. Often the anticipated performance is dependent on fast hardware, although good algorithms are of considerable assistance. The designer's job is to obtain a sufficiently good representation of the plant dynamics, and to insure a response to given excitations in the predetermined time by the appropriate choice of fast hardware and software, based on the cost constraint. Most plants also have a human operator in the control loop.

The basic task the Operator is to maximize the cost-effectiveness of the production process while minimizing the likelihood of damage to the plant and personnel in the event of component failures and/or other events not included in the control algorithm. The operators must, therefore, attempt both tactical and strategic control, depending on the complexity of the control system design. The functions of the controller and the instrumentation system are the following:

Tactical;

- To reduce the need for operator intervention by automatically controlling the plant (i.e., to implement tactics).
- To reduce the risk of damage to plant and personnel by automatically shutting down part or all of the plant when process variables move out of their normal operating ranges (i.e., failure and/or disaster containment).

Strategic;

- To provide up-to-date information to enable optimum control strategies to be implemented by the operators.

The implementation of an appropriate strategic response to an emerging situation depends on the experience of the operator and on the ability to recognize and respond appropriately. In addition, the current cost/performance and flexibility provided by process-control computers means that it is possible to present to the operators a large quantity of information. The operator, in such situations, must respond to this information and must often make complex strategic decisions. As a result, when things start going wrong, accompanied usually be a sudden proliferation of alarm information, both the operator's ability to respond appropriately is stressed resulting in errors of judgment, and response time limits are often exceeded. This phenomenon is known as 'cognitive overload'.

There is, therefore, a need, in many applications, for systems capable of reducing the cognitive load on operators by focusing on underlying plant problems and automatically implementing correct tactical and strategic functions. In extreme cases, there is a need for eliminating the human operator altogether. These requirements suggest the use of knowledge-based software (often called an expert system).

The representation of a plant by knowledge-based software is appealing from several points of view. Often the plant has many features, which are difficult to model by ordinary means, and such features are often easy to state as rules. The representation of a plant by a knowledge base is a simulation of the plant, which can be based on compiled knowledge, as well as on heuristics obtained from observed behaviour. These are often easier to represent by rules, based on knowledge, than by attempts to model it mathematically. A knowledge-based representation maintains all the attributes of extensibility, transparency and simplicity normally obtained from these systems.

In addition, strategic approaches to control can be instituted easier than in classical approaches. Smart controllers are possible, therefore, not only in a tactical but in a strategic sense as well. Thus, complex inferencing about control options can be instituted for such considerations as failure containment, emergency response tactics, and responses to developing pathologies. In addition, the operator's tactical and strategic role can be reduced (thus reducing the stress level) and/or eliminated.

Finally, it is often possible to get a working system with a higher performance-cost ratio than by classical means. There are, however, problems in both the design and implementation of this approach: primarily in insuring a good representation of the plant dynamics that is executable within the time constraints.

This paper provides a state-of-the-art look at progress in the utilization of expert systems technology in the design and control of real-time expert systems. It provides a report on the published work in the field, as well as who is working in the area. It discusses the design of real-time expert systems, the problems involved in developing them, and the approaches taken to overcome the obstacles. It thus serves as a first point of reference for anyone considering work in the field.

1.0 INTRODUCTION

The descriptive phrase "a real-time system" is used widely by various design teams to describe very different performance attributes. On the lower side, 'real -time' is often used to mean 'fast', in some vague sense, either relative to an external environment or in terms of computing resources consumed to produce the response. For our purposes, a real-time response forms part of the design requirements and is a constraint which must be met before acceptance. Regardless of the algorithm, the system must guarantee a response to a given external stimulus in a given time (most often, the results of not meeting the time constraint is catastrophic rather than merely inconvenient). The set of stimuli, and the assorted set of responses form the real-time constraints on the design.

More formally, given a set of inputs §S† and associated set of responses §R†, and the associated response time intervals §T†, the designer must guarantee that after an input Si, the elapsed time T before a suitable response is generated is such that T is less than Ti.

In some situations, the quality of the response will improve given more computational time. In such situations, an adequate response may be computable within the given Ti, but not necessarily the optimum. A carefully defined adequate solution may form part of the design requirements. The designer's problem, in this case is, to organize the knowledge base and inferencing algorithm so that the quality of the solution improves monotonically with the search time and so that the solution found in Ti is always acceptable.

The representation of the control characteristics of a plant can be undertaken by the functional partition of classical expert system software i.e., a rule base and an inference algorithm. The input excitation from the plant is intercepted by the inference algorithm and the search for a suitable output excitation involves finding the consequence of the appropriate rule. The problem has many facets: the most obvious being to guarantee the response time (i.e., the search time of the knowledge base) based on the specified constraints. The response required of such systems could either be the best, within the time limits imposed, or the optimum obtainable, within the same limits. Each requirement poses special design problems.

In most applications, in order to achieve the benefits of this technology, the knowledge base is large and complex, and the inferencing techniques are, therefore, subject to combinatorial explosion problems (the search tree growing exponentially), and sometimes a version of halting problem, in which the search is not guaranteed to terminate [28]. Several approaches to speeding up the search have been proposed (these are, of course, applicable to regular expert systems), which involve compacting the knowledge base and partitioning it so that only the most probable partition is searched.

Implementation of the appropriate compacting and partitioning approaches would reduce the execution time and possibly allow implementation of a real-time response: however, 'fast' is not necessarily real-time, and many additional techniques are usually adopted to guarantee appropriate response times.

Finally, knowledge-based systems can be designed and implemented to respond to input data which is incomplete and/or uncertain, by utilizing reasoning processes that acknowledge uncertainties. Reasoning with uncertain data is an attribute that is relatively easy to implement using a knowledge based system.

This paper discusses how these approaches have been applied. It reviews real-time expert systems implemented to date, as well as the problems that have been encountered in developing these systems, the design approaches that were used to overcome the problems and the further issues that need to be resolved before these systems may be widely used.

Section 2 provides a discussion in more detail of the requirements for real-time systems that apply generally, regardless of the implementation.

Section 3 is a summary of the features of several systems, which illustrates both some typical applications and the various approaches used during implementation.

Section 4 reviews the approaches, which have been published to date, aimed at addressing the issues and for solving the problems.

Section 5 is a perspective of the major problems that have yet to be solved before the wide-spread use of this approach will be viable.

2.0 REAL-TIME REQUIREMENTS

A number of issues beyond those usually considered in classical expert systems design must be addressed before choosing such systems for real-time problems.

Obviously, in order to guarantee the real-time constraints, the system must provide the response to changes in the environment at or before the time it is needed. Achieving the design specifications requires, as usual, a judicious allocation of the response-time budget among the instrumentation, the processing hardware and the software. The software is our main concern here, and this software can be a hybrid of classical (i.e., algorithmic) and knowledge based modules. The design of the algorithmic software follows all the rules, which are now well understood. The knowledge-based portion of the software is the novel part and is of the most interest in this paper. The obvious first approach is to reduce the size and complexity of the knowledge base and to find high-speed inference procedures to search for responses. This may not be easy, or even possible, since large and complex representations and inferencing procedures may have motivated the use of this technology in the first place.

The results of the designer's efforts may be thought of in three catagories: representation, functional allocation and performance:

Representation: Representation is thought of as encompassing both the logical representation of the plant dynamics and its operational requirements (both tactical and strategic) and the implied inferencing procedures. The representaion used has been either rule-based and/or frame-based systems.

Inferencing procedures are often the most difficult to model and to implement. Both forward and backward chaining are used as are many variations of hypothesis testing and truth maintenance.

Functional Allocation: The allocation of functions between hardware and software is dependent on the performance/cost ratio that must be achieved.

The first major decision with respect to software is the allocation of functions between classical algorithms and the knowledge-based software. Normally, the operational attributes allocated to the knowledge-bases should be those that are difficult to represent by algorithms, both in the representation of the plant dynamics (i.e., tactics) and those used to implement strategic operations or responses based on experience and/or heuristics.

It is a truism, perhaps, that each functional part of the controller should be allocated to the mechanism that most cost-effectively implements the requirements. This implies that a good design methodology is available to expose the functions and to partition the design so that a proper allocation can be found from the alternatives. This is a separate topic for discussion, which is beyond our scope.

Performance: Achieving real-time performance within the cost constraints is the bottom line from the designer's point of view. It involves both achieving a good representation and an appropriate functional allocation. In addition, it almost always involves choosing fast hardware as execution vehicles. The first and most important step is to create a functional architecture that allows optimum decisions to made with respect to the partition and allocation to implementation hardware. This implies that the software/hardware boundary can be chosen with equally well.

Given that a good functional architecture has been chosen, the task can be considered in two parts: first, the intrinsic response obtainable from the software and second the speed of the hardware portions. It may take many design iterations to find an optimal solutions as well as testing and tuning during the field trials.

Parallel processing becomes an important consideration with the implied requirement that tasks can partitioned and allocated to take advantage of the inherent concurrency. Once again we are back to the need for a good design methodology.

3.0 Application Examples and Approaches

In order to illustrate the benefits from an expert systems approach, it is useful to evaluate the problem domains or areas that have already been tried, and the level of success of the systems built to date. This section is divided into two parts: Section 3.1 outlines a generic overview of some problem areas that have benefited from implementing real-time expert systems; Section 3.2 contains an extensive discussion of systems that have been reported in the literature as either working or in progress.

3.1 Domain Areas:

The following represent some of the generic applications reported in the literature. They illustrate a grouping of the various domains and illustrate some of the specific tasks allocated to these systems.

INDUSTRIAL

1) Process Control

a) Process upset, i.e. handling of the alarms.
b) Diagnosis of inconsistent information (for example, critical measurements that do not result in an alarm).
c) Strategic economic optimization.
d) Provide strategic advice to the operators to help them handle and avoid crises.

2) **Maintenance Tasks:**

Assist the smooth running of the plant by providing a quick diagnosis of the fault and recommending a quick repair or advising on the most suitable action. These expert systems are best regarded as an advance in the automation of the factory They perform the following tasks:

a) Alarm monitoring [ES for maintenance tasks].
b) Fault diagnosis.
c) Test generation.
d) Automatic plan rescheduling.
e) Sequencing of repair action.
.) Management of the scheduling and long term planning of the plant's operation.
h) The entirety of the plant may also be kept in perspective, so that the propagation of faults throughout a plant may be detected.

3) Computer Operations Environment:

A continuous real-time expert system for computer operations.

a) Controlling the computing system.
b) Provide real-time advice for the operators.
   1) Scheduling large batch of jobs.
   2) Detect and suggest correction for hardware errors.
   3) Performance monitoring.
   4) Background monitoring (to verify all parts of the environment are operational).

MILITARY

a) Autonomous satellite that participates in early warning missile tracking and counter measurements.
b) Satellite self defense activities.
c) Nuclear power plants.
d) Smart weapon systems.

MEDICINE: Intensive care patient Monitoring

COMMUNICATIONS: Message interpreter (for example, for ships and airplanes).

3.2 Example Systems:

The following examples provides a description of several real-time expert systems that are either in routine use, in field testing, or in a development environment. The goal is to provide a comprehensive view of the variety of systems and where posible the approaches used in resolving the various issues faced by the designers.

HEXSCON: An Expert System for Real-Time Control [26]

Problem Domain: Hexscon was developed for military applications which demanded high degree of sophistication with strict constraints.

Design Goals:

1) Accommodate 5000 rules in a microcomputer system with 512 k memory.
2) Response time of 10 ms to 100 ms.
3) Ability to handle many objectives (about 1000)
4) Ability to function with uncertain data.

Architecture: Uses both conventional logic and knowledge based techniques, to preserve adequate and rapid operational decision making.

The problems that arose from this approach was the need for compatible knowledge representations in the two parts. Therefore, to make a completely integrated hybrid system, knowledge and data representations for both conventional and knowledge-based parts would have to be identical or at least fully compatible. The solution was to use a representation translator. This allows the conventional and knowledge-based parts to exchange information freely without requiring full representational compatibility.

Knowledge Representation and Inferencing: Real-time systems are usually partitioned into highly modular tasks, communicating under control of a real-time operating system. However, it proved difficult to implement the knowledge-based part of the system by partitioning it into highly modular tasks due to the following reasons:

- Forward applications of conventional real-time techniques were extremely expensive in scarce memory resources.

- A sophisticated partitioning of the inference engine introduced problems in implementing efficient knowledge representations that had been developed in stand-alone knowledge based systems.

The solution was to let the knowledge based portion run as a single task in the real-time system using a forward chaining (data driven) inference engine. Production rules are used for representing the heuristic knowledge of the problem domain.

Language: At the beginning of the project LISP was chosen but it was abandoned since LISP languages for microcomputers tend to be memory intensive and execute significantly slower than other microcomputer languages.

Pascal was then chosen because it simplified construction of large programs, produced compact, fast code in the target environment, and it could be easily converted to Ada.

Status: A prototype real-time knowledge-based system was developed on a microcomputer.

Performance: The response time varies from 0.10s to 30s depending on the computer, and the number of rules used in the knowledge base. The response time was 0.25-0.5 s with the emphasis on maximum speed, and 1 to 10 s with the emphasis on maximum expertise.

Assessment: Although the system does not achieve the ultimate goal for the response time (10 to 100 ms) for most cases, it shows that a knowledge based controller could be built that runs on a microcomputer and has enough sophistication and capability to be effective for some real world problems such as military applications. It also shows the state of the art in predicting expert systems performance.

ESCORT: An Expert System for Complex Operations in Real-Time [24]

Problem Domain: Escort's knowledge covers instrumentation failures and some operator errors. Plant failures, such as pipe rupturing, were excluded because of the relative rarity of such problems.

Design Goals: The design of ESCORT was influenced by the following factors:

1) The system must provide advice on plant crises within 1 second, (a requirement typical of conventional process control systems)
2) This advice must both indicate the cause of an alarm and the importance of the underlying problem relative to other existing problems.
3) The system should provide advice on control and instrumentation problems but not plant failures.
4) The operator must always remain in control of the system; not the other way around.
5) The delivery system should be capable of installation without modification to the target plant.
6) There must be a net benefit to the operator of using, rather than ignoring Escort.

Architecture: Escort has two user interfaces, the first provides continuous advice on plant status to the operator; the second enables enhancements to be made to the knowledge base.

The user interface was considered to be critical in the design because of the following reasons:

1) It would be a major factor in the usability of the system
2) Its impact on the Escort's knowledge and reasoning approach.

These resulted in the following requirements:

1) Advice on the problem causing alarm
2) Explanation of the basis for advice
3) Indicate the relative importance of the alarms
4) Indicate the likely affect of not dealing with the problem
5) The operator should always have control over which advice he is given.

The following are the tasks that Escort needs to perform:

1) Recognizing occurrences or events in the processing plant which may indicate some plant problem or operator error.
2) Prioritising these events.
3) Analyzing an event to infer the underlying plant or instrument problem or operator error
4) Prioritizing the underlying problems
5) Presenting problem diagnoses to the operator
6) Responding to operators request.

Of these, all but the last two use knowledge-based approaches.

Knowledge Representation and Inferencing:

Time critical systems require more sophisticated ways of determining reasoning strategy than using explicit representation of domain knowledge. This is due to the fact that they have to cope with widely varying computational loads, deal with multiple events in parallel, and respond quickly to events.

Escort, therefore, uses a knowledge based scheduler. It provides the flexibility to allocate resources depending on the current state of its problem solving and of the plant. The knowledge-based scheduler is able to take account of the plant state, the diagnoses made and any operator requests in determining which activity to perform next.

The scheduler is responsible for ensuring that the individual tasks and in particular the diagnosis task are performed efficiently. For the knowledge-based tasks, this is essentially a search problem in which a solution is obtained by examining and applying as few rules as possible. Rules applicable at similar instances or at similar stages in the diagnosis task are therefore grouped together into rule sets so that, at any stage in the problem solving, only potentially relevant rules are examined. This has meant that typically 20% of the rules are activated in diagnosing a particular problem.

Assessment: The system is capable of reducing the cognitive load on a plant operator. However, there is no mention of the systems response time in this paper.

Future Work: The recent work has concentrated on making Escort's diagnostic reasoning more flexible. Escort will be able to follow the cause and effect pathway from the original problem to its manifestation elsewhere in the plant using the knowledge-based search strategy. The new approach will also enable Escort to reason over time rather than just within time.

YES/MVS: A Continuous Real-Time Expert System for Computer Operations [22]

Problem Domain: The requirement for high availability and performance in large computing installations has increased the need for fast and consistent responses to operational problems. Therefore, automatic aids for computer operators are needed.

YES/MVS is an experimental facility developed to aid in the real-time operation of a large multiple virtual storage/system Product-Job Entry Subsystem 3 (MVS/SPJES 3) computing installation. It addresses both routine actions taken in operating the target system and spontaneous problems that would normally be handled by an operator.

Design Goals: The problems addressed include:

1) Various kinds of MVS-detected hardware errors.
2) JES queue space depletion.
3) Channel to channel link problems.
4) Subsystem abnormal ends.
5) Performance monitoring.
6) Background monitoring.

Problems detected and positive corrective actions are displayed for the operator.

The following benefits were anticipated:

1) Better management of the different processes which are running asynchronously as part of YES/MVS.
2) Relieving the expert system from low level input/output considerations such as message and display screen formatting.
3) Providing a self contained knowledge base, so that operational policy description can be more easily read and modified.

Architecture: YES/MVS runs on a separate computer and does not depend upon the target system for computing time and other resources in order to be able to handle major incidents in the target system. Its interface to MVS is through an emulated JES 3 console, appearing to MVS as a normal operator's console. YES/MVS runs in three concurrently running virtual machines under VM/SP operating system:

1) The expert virtual machine.
2) The MVS communications control facility virtual machine.
3) The display control virtual machine.

The expert virtual machine: Using the expertise from multiple subdomains together has several advantages over creating a separate expert for each subdomain in its own virtual machine. Since rules in a given subdomain may use some of the same status information that is needed by the rules in another subdomain. One global model of the target system is kept in working memory. This eliminates redundencies and, hence, inconsistency across subdomains in the expert system's model of the target system.

The MVS communications control facility virtual machine: It runs the MVS communications control facilities. This virtual machine frees the knowledge engineer from concerns about parsing messages and extracting internal character string.

The display control virtual machine: It controls the display facilities and all interactions between YES/MVS and the systems operators. The actual implementation of display facilities was carried out through the use of an OPS5 rule base, combined with calls to the IBM Graphical Data Display Manager.

Knowledge Base and Inferencing: The knowledge base is a high level, declarative format of production-rules. Since the knowledge base is inherently modular, it is easier for it to adapt and evolve with a procedural encoding of the knowledge.

Language: OPS 5 was selected as a vehicle for implementing YES/MVS due to its several advantages for the project.

1) It is flexible and modifiable
2) It could be converted to run under LISP/VM on an IBM computer
3) Its use of production rules as a knowledge base representation seemed naturally suited to the type of knowledge occurring in the domain of computer operations.
4) Its data driven form of inference was particularly appropriate to the situation of responding in real-time to information received from the target MVS system.

However, significant extensions were made to OPS5 to enable it to be used in a real-time application. The expertise in YES/MVS on system operations is encoded in an extended version of OPS5, which runs within the LISP/VM environment.

The Requirements for Real-time Control: The MVS system being monitored and controlled by YES/MVS is highly dynamic. Since the MVS world is highly nonomonotonic, it is impossible to maintain an accurate model of MVS that is complete in all detail. Instead, a model that provides a reasonably good description of the status of MVS, from the view point of operations is maintained. The various problems discussed earlier were addressed as follows:

Problem: Real-time response.

Solution: The speed of execution of OPS5 was improved by compiling the right hand side of the rule (such a compilation has been independently introduced in OPS83). The matching process has been tuned with several LISP macros. Also the rules were distributed between several OPS5 systems using concurrent processes in the form of separate virtual machines supported by a host computer.

Problem: Being able to initiate an action at a given time is one of the fundamental requirements of a real-time control problem. With a data-driven inference engine, this includes the production of working memory elements at some future time.

Solution: This was accomplished by defining a new right hand side action primitive for delayed production. TIMED-MAKE, which takes the normal OPS5 MAKE arguments followed by either an absolute or relative time specification.

Problem: The ability to have distributed processes interact in a timely fashion.

Solution: Fast communication is achieved by introducing a new communication phase in the normal OPS5 inference cycle (recognize, conflict resolution, act). During the communication phase, external messages are picked up and outbound messages are sent. The conflict resolution then takes place based on changes to working memory as the result of both right hand side actions and incoming messages.

Problem: Need for explicit control. For example, there are critical problems that require a command sequence to be issued to MVS without other queries or commands being interspersed. Hardware error message handing is one such case. Such a real-time requirement necessitates explicit control over the rule firing in the inference engine.

Solution: The two modes of OPS5 conflict resolution, LEX and MEA, were extended by a Priority Mode which has precedence over these. The conflict resolution of OPS5 is modified so that the active conflict set is temporarily reduced by excluding all active rules that do not have the highest priority task among set. Then the normal OPS5 conflict resolution acts on the reduced set.

The priority mechanism effectively satisfies the real-time control needs. It also allows control over rule interaction between different subdomain areas. While meta-rules could have been used to refine the conflict resolution strategy, it was found that the priority control mechanism offered an equivalent capability without incurring the overhead and complexity of such a facility. Furthermore, it provides a rule-grouping paradigm similar to the use of contexts in EMYCIN and rule-groups for HH rules in EXPERT.

Requirements for continuous operation: There are at least three basic requirements for operating in a continuous mode:

1) The inference engine should not terminate when no rule is eligible to fire.
2) The system should ideally run on a special-purpose, high-availability computer, different from the subject machine. If the host computer or the virtual machines comprising the system go down, the system must be restarted.
3) Working memory elements that have served their purposes must be removed. The accumulation of old useless data in the working memory not only creates memory space problems in continuous operation,but, of more importance, instantiates the wrong productions in a data driven inference engine, such as OPS5.

In order to overcome these problems the following steps were taken:

1) An OPS5 rule OPSWAIT was implemented, which puts the system into a waiting mode. Any external messages causes the system to resume, with the new data added to the working memory.

2) An automatic restart procedure is called during the host computer initial program load and also when a down machine is detected during a periodic mutual polling among virtual machines of the system.

3) Many different "garbage collection" techniques have been used to remove old data.

Language: OPS5

Status: YES/MVS ran regularly at the Yorktown Computing Center during a period of over nine months. After considerable effort, it ran fully authorized, taking action automatically and subsequently notifying the operator.

Assessment: The system certinly worked as designed, however it is interesting to observe the continuous increases in cpu memory being used by the monitor. The cost/performacne is certainly something tht will have to be looked at from several points of view.

Future Work: A second version of YES/MVS is being developed by the Expert Systems for System Management Group. This second version wil` ·e installed at several large IBM computing centers. It will contain software to address problems that are sufficiently uniform that generally applicable knowledge can be used to control diagnosis and corrective actions. It will also emphasize software constructs that support the natural statement and direct implementation of knowledge about the operational policy that is unique to a particular computing installation.

IAP - Feasibility Study For an Energy Management System Intelligent Alarm Processor: [21]

Problem Domain: Modern energy management systems all have some form of alarm processing to alert the operators to power system parameters that are out of normal range or to changes that may affect the operation of the power system. Alarms can be processed and given to the operators on the CRT display very rapidly and this has led to concern about the way alarms are processed.

Design Goals: Several recommendations for improving the alarm processing has been made. The following are a few examples:

1) EMS systems should present transformed data to the operator not just volumes of raw data.
2) The transformation of the alarm data should be done in such a way as to improve the operator's view of the power system, offer a larger and better base to make decisions, and convey a clearer idea of the power system condition causing the alarm.
3) Alarm priority should be changed dynamically as system conditions change.

Architecture: One assumption build into the Intelligent Alarm Processor (IAP) is the need for processing speed so that the expert system does not itself put an intolerable delay in the display of alarm results and analyses. For this reason, the rules that govern the processing of the alarms can be set up to reduce the depth of analysis performed on each alarm when a large burst of alarms is encountered and then to return to the normal analysis after the disturbance has passed.

Language: The first several prototypes of the expert system were written in FORTRAN since most power systems analysis software is written in FORTRAN the people trained in its development are usually well versed in this language. However, later in the design process LISP was found to be a superior language for developing the system. The only drawback to using LISP is the need to occasionally call for results from a calculation which exists in FORTRAN program. Therefore, future implementation of expert systems in EMS systems will require one of the LISP systems which can interface to other languages when needed.
Status:

Assessment: The IAP appears to be just a prototype and it still has a long way to go before it could be used in the EMS systems. Its feasibility was tested using a dispatcher training simulator and it has not been tested on the real modern energy management systems. The was no indication of the response time of the system.

Future Work: The work so far has demonstrated the feasibility of using AI techniques to solve a very difficult problem facing the users of energy management systems. More extensive work is now being conducted to determine the type of rules and the ways in which the knowledge base should be built for an actual IAP installation. It is expected that a working prototype will be tested in an EMS installation by 1987.

PICON: A Real-Time Expert System For Process Control: [9]

Problem Domain: In a large process plant, such as a power plant, refinery or other such process, there are several thousand measurements and alarms provided to the human operator. The large size and dynamic nature of the domain requires new approaches to the inference, since exhaustive search procedures are not possible in real-time.

Design Goals: Design an expert system to perform inference as would a human expert, who is confronted with the same problem of a limited time to respond to complex situation. The key concepts are:

1) To recognize quickly process conditions which are potentially significant; and
2) To invoke relevant rule-sets and focus on these problem areas for diagnosis and procedural advice.

In addition, there were two real-time design considerations:

1) Dynamic nature of the domain "facts" presents a particular challenge.
2) Large size of the knowledge base required for a realistic implementation.

Architecture: The Process Intelligent Control (PICON) package is designed to operate on a LISP machine interfaced with a conventional distributed system, where as many as 20,000 measurement points may be assessed. This presents a major communication problem, which was solved by using the LAMBDA machine from LMI. The LAMBDA provides two processors running in parallel: a LISP processor for the expert system and a 68010 processor for real-time data access and certain low-level inference tasks.

Considerational efficiency led to a design utilizing two parallel processors:

- A 68010 running a C-coded package called RTIME (Real Time Intelligent Machine Environment). RTIME performs process condition analysis, as directed by the expert system in a LISP processor.

Logic rules an procedures are used when required for the diagnostic inference. PICON mimics the expert process operator in this regard: logic rules and procedures are invoked specially when they are required for diagnosis of a process problem, or as requested for a specific step in the inference.

The design includes the ability to change the time period of measurement and algorithm processing in individual cases. Therefore, in effect, the system can "focus attention" to a specific area of the process plant, and put all associated measurements and rules for that area on frequent scan. This can be done under control of the LISP program.

Another use of this "focus" facility is to scan the plant in a background mode, focusing attention on parts of the plant to evaluate unit process performance and detect subtle problems, utilizing both the knowledge base of the expert process operator and the expert process engineer.

It should be noted that the ability to focus not only emulates the way a human expert works, but also it avoids the problems associated with overloading the distributed process system with request for information.

Knowledge Base and Inferencing: The inference engine includes both forward and backward chaining, since within the context of an alarm advisor, there are requirements for both of these procedures.

There are two types of knowledge in the knowledge base:

1) Process control knowledge
2) Heuristic knowledge

The heuristic knowledge base is organized in a taxonomy of frame-like structure. This was chosen for generality, and also to facilitate explanation. It includes provision for certain factors, permits backward chaining diagnostic inference.

Status: The PICON package provides a knowledge based structure, facilities for acquiring the knowledge base in an organized manner, and real-time collection of data with some parallel processing of inference,and higher level inference tools. The individual applications require specific knowledge engineering, which is facilitated using the tools we have described.

Future Work: In the future applications such as plant optimization, overall plant and corporate scheduling, plant design and other high level planning will be considered for expert system application. These will require further development of yet more sophisticated expert system tool.

A RULE-BASED SUPERVISORY CONTROL SYSTEM FOR MANUFACTURING: [19]

Problem Domain: The cement manufacturing process, and in particular the kilning stage, has acted as a focus for the assessment of alternative control techniques, and specially for the investigation of rule based control.

There are two main reasons for this attempt: first the kiln process is very complex and is subject to a number of problems such as, changes in raw materials composition, random process disturbance, and long process lags. Past attempts at automating the supervisory control of the process based on conventional techniques have not been sufficiently robust to be useful in the kiln environment. Secondly, the potential payback in terms of energy saving is very large.

Architecture: The computer system is based on the PDP 11 series (11/23 or 11/73), with all the software running transparently under the RSX-11 M operating system, providing a full multi-tasking multi-user real-time environment. All software is sourced in C and MACRO; conventional AI languages were considered but discarded at the system design stage, due to the real-time operating constraint and the amount of floating point mathematics required.

The number of data base "items" that can be accommodated is dependent on the memory available.

Knowledge Base and Inferencing: Data frames are used to describe plant items and their treatment. Several types of data frames are available, which in effect form a composite knowledge base.

Inferencing procedure: Knowledge, strategy, procedure and control mechanisms are all handled by the Linguistic Control Language, a language with limited forward chaining combining the normal functions of real-time process control languages with a technique for embodying knowledge of the process expert as sets of linguistically expressed rules. The language also handles uncertainties.

Status: The system has been installed in Blue Aberthaw works since early 1985, and has been conducting the closed loop supervisory control and optimization of the process. In use, the system has proven its worth. Linguistic rules have been shown to permit complex processes to be easily described and successfully controlled. It enabled the Blue Circle to gain ever greater insight into the process. Finally, all the indicators are that the 5% energy reduction aimed at is achievable, with all the cost saving implications. The system is being jointly marketed as a commercial product.

AN EXPERT SUPERVISOR FOR A ROTARY CEMENT KILN: [14]

Problem Domain: Conventional control schemes in cement manufacture have not succeeded in achieving the desired results due in part to both cost and process constraints. Processes such as this are generally controlled with a significant level of human experts.

Knowledge Base and Inferencing: The knowledge base has been developed in the form of production rules. The operating mechanism for the production system consists of a long term memory (LTM) and a short term memory (STM). The former contains set of rules and the latter data pertinent to the current state of the process. When information in the STM has been utilized it may be replaced in total or in part by new data. The main program is goal driven, with top goal being to determine the KILN CONDITION.

Language: The logical language PROLOG was chosen for writing the sets of rules. The version used was YORK Portable Prolog for which source code in ISO Pascal was available. This has been installed on several computers, such as IBM-PC.

Status: This paper only presents an approach for developing knowledge based process supervision and control for cement kiln. The author claims that the techniques for developing expert systems in this situation are not yet well established. The total knowledge is neither available nor capable of complete utilization within the present structure for knowledge representation.

THE ALVEY RESCU PROJECT- A PROGRESS REPORT [20]

Problem Domain: RESCU is a real-time expert System Club of Users who are collaborating with financial assistance from the ALVEY program in a project to establish the potential of exert systems in process engineering.

Design Goals: The application selected was to apply the expert systems techniques to the quality control of an Ethoxylates plant at ICI Wilton. The system covers process modeling, process tracking, control, recipe, recommendations and explanations.

Architecture: The expert system is developed on a DEC VAX 11730 under VMS using the POPLOG development environment, with a color display system used to support the plant operator's interface and plant data being acquired automatically from an already installed Foxbourough Fox 1/A computer.

Language: Poplog

Status: May 1985, the project moved to the prototype development and integration phase. At the end of August, nearly all components had been completed and some had been tested in isolation. Some components had been combined into subsystems and tested as cooperating elements under communication infrastructure.

Future Work: The next stage of the project is to move towards complete prototype integration in parallel with process of on-site knowledge elicitation and representation on the RESCU KRL. There will then follow a period of on-site commissioning, knowledge refinement and assessment of the prototype.

INTENSIVE CARE UNIT MONITORING USING A REAL-TIME EXPERT SYSTEM: [2]

Problem Domain: Conventional alarm systems are typically unable to handle patient/disease specificity, temporal/dynamic changes,and multivariate interactions. The existing alarm systems used in monitoring ICU have many limitations. In actual clinical practice, the good clinician is able to circumvent these limitations. But, what if an astute clinician is not attending the patient? Under these conditions, it is quite possible that a significant physiologic derangement is missed. Therefore, some means is required to help the clinician in these situations. A real-time expert system was proposed to do the job.

Architecture: Production rules are proposed for developing the system. The knowledge base is divided in to chunks of rules were each node contain production rules that are applicable during a certain time interval. That is, these chunks of production rules could be viewed as separate systems which present distinct processes. Therefore, this architecture is fundamentally parallel in nature. The system is designed to accept concurrent, asynchronous inputs. Since the processes are not independent of each other, there must be some method of interprocess communication.

Strategies for enabling communication between the various production networks is being examined. The following are the issues that must be considered:

1) Certain types of information have to move more quickly through the hierarchy than others.

2) The system must be able to handle potential conflicts, for example, two conflicting expectations.

It is not clear whether conventional communication techniques are powerful enough, or even appropriate, for the needs of the system. These standard techniques have arisen in traditional algorithmic environments, not heuristic ones. Moreover, the techniques are algorithmic rather than heuristic.

Status: To date the project has accomplished the following:

1) Identified certain major weaknesses in conventional alarm systems;
2) Developed a novel production system architecture that is designed to alleviate these weaknesses;
3) Written many of the rules for individual parameters
4) Begun to consider lower level implementation details.

ERIK - An Expert Ship Message Interpreter: Theoretical and Experimental [25]

Problem Domain: The interpretation of communicated messages become exceedingly complex due to ill-formed constructs and noisy transmission channels and requires a considerable amount of human information processing.

Design Goals: Make use of the available technology for processing natural language.

Interpretation Requirements

1) Considerable amount of task specific knowledge
2) Large degree of flexibility in processing
3) Efficient error recovery mechanism.

Processing requirements:

1) Interpret or reject a message in less than 60 seconds of processing time.
2) Must be able to process correctly at least half of the incoming messages.

A failure to meet any one of these requirements would have drastically reduced the practical validity of program. The following processing difficulties and goals have guided the major design decisions in creation of ERIK:

1) Boundaries between reports on multiple messages have no clear markings. Breaks between reports must be recognized nevertheless, or else an entire report may be parsed incorrectly.
2) The reports are often ill formed.
3) Due to time and accuracy constraints, ERIK must have the ability to quickly identify cases that it cannot process and, consequently, to terminate further processing. However, the rejection of reports must, in general, be kept to a minimum.
4) Only a small fraction of the entire information in a message is relevant to ERIK. Determining the relevant parts is one of the major concerns.
5) No string dictionary exists for many fields, making identification of those fields difficult.
6) A limited natural language capability should be given to the interpreter. This is because natural language is occasionally used in reports that describe ship sail plans.
7) The access to the data base containing the ship names and call signs should be held to minimum, since it is very large.

Architecture: ERIK runs on AMVER (Automated Mutual-assistance Vessel Rescue). The source code in LISP is 541 k words. It takes 1218k words in dynamic memory of common LISP (this includes the index into the ship and port name data base but not the data base itself).

Language: LISP.

Perfromance: Average processing time per ship message depends on the hardware on which ERIK runs. For example; on DEC VAX 11/785 running common LISP:

the average CPU time per ship report	= 19.35 seconds,
the average CPU time per message	= 27.5 seconds, and
the average CPU time including LISP garbage collection time	= 25.51 seconds per ship report
the average CPU time including LISP garbage collection time	= 36.26 seconds per message

However, when running on Symbolic LISP machine, the equipment used by the Coast Guard, these numbers are reduced by a factor of two.

Future Work: It includes studying aspects of ERIK that are cognitively valid on tasks such as expectation-based spelling correction, which is integrated with conceptual analysis, and efficient interpretation of garden path sentences.

4.0 UNRESOLVED ISSUES AND PROBLEMS

The systems described in Section 3 are not yet exploiting the full potential of this technology. In the following several issues are discussed, which form the concern of researchers in the field, whose resolution will hasten the advent of large scale real-time expert systems. The issues range from the ability to think about and represent strategic requirements to the problems of knowledge representation and inferencing already discussed.

Strategy and Inference Procedures: Current real-time expert systems tend to use heuristics to guide the search at a surface level, and for relating observed signals to implied underlying causes and faults.

Few applications so far have considered the dynamics of the systems they are monitoring - most look for combination of static signal conditions. For less complex applications, these restrictions are often tolerable, but future controllers for very complex systems, capable of reasoning about and controlling new situations, will have to reason at deep levels about cause and effect and dynamic behaviour of the observed system. Two areas of interest become important: deep knowledge and simulation as a means of inferring future behaviour.

If deep knowledge could be used both to reason about and to control new situations, then simulation could play a very substantial role in the control of a system. Simulation could be used as means of inferring future behaviour and checking the credibility of the hypotheses.

It is possible to reason about different properties and constraints than those presented in the knowledge base only if we use methods of reasoning at the level of general cases and theories. The heuristics are then at a much deeper level than surface heuristics. There may be some drawbacks, however, since these levels don't encompass any of the induced heuristics that an experienced plant operator may have gained. Therefore, there may be different levels at which the knowledge-based control could act. The higher level representations are more flexible, but are more expensive in computational terms. However, many are convinced that by concentrating efforts on systems working at these high levels, more intelligent systems can be built. In particular, we may be able to construct systems which can reason correctly in a certain percentage of unforeseen circumstances; a rather exciting possibility.

Representation: The success of a rule-based controller depends obviously on the representation of the process. The rules need to be compacted, if the speed of response of the system is not to deteriorate as the number of rules increases. This may be done in several ways, for example by grouping together the rules which apply to a particular component, or by ordering them according to the number of separate clauses in the antecedent of the rule.

A frame-based structure lends itself particularly well to process control and maintenance. A frame plant component' may be defined, describing basic properties. Particular kinds of component are defined as subclasses. The advantage of this kind of knowledge representation are flexibility and capacity to present °deep' knowledge about physics and materials.

Deep level representation of knowledge is more expensive in computational terms. However, many are convinced that by concentrating efforts on systems working at this level, more intelligent systems could be produced.

Reliability and Sanity: It appears that none of the systems built so far have attempted to provide means for checking the knowledge base for inconsistencies. If a procedure could be developed to check this, then it would reduce the search time for the cases in which inconsistencies occur and hence reduce the search complexity.

5.0 CONCLUSION

Knowledge-based systems are now being used, and will become more wide spread as more experience is gained. An expert system, none-the-less, is quite similar to a real-time control system; for example, both are command and event driven, have feedback loops, require the same instrumentation packages, and access the same kind of data from conventional data bases. The total software package is often hybrid in nature combining classical algorithmic representations with knowledge bases and inferencing procedures. The designer must allocate both the functional tasks and the response-time budgets among the various implementation devices. Special purpose hardware and software for this application is often essential.

The systems presented here have dealt with most of the real-time requirements. It seems that they all have avoided the problems encountered when attempting to deal with dynamic facts by treating them as static facts for a particular instance. However, this approach can not be used for more complex systems, and it does not provide any insight into the past history or future expectations. Therefore, none of these systems can predict future problems. An approach for solving the °dynamic facts' problem is to employ deep heuristics or knowledge by applying qualitative theorems. The area of qualitative heuristics requires more research at the present time.

Setting up an Expert System:

For an on-line system attached to a control system, a good approach would be to let each system do what they do best: a classical control system should handle real-time data based on algorithmic tasks, and the rule-based software should perform the rule-based representation of the plan dynamics and the strategic tasks. The issue is how do you arrive at the optimal partition within the cost/performance window.

LISP for real-time control operations:

Any language or development system that uses linked lists and heap storage, must periodically engage in a process of compacting memory storage areas that has become deallocated. This proces is called "garbage collecting". A LISP based system, for example, must execute such a procedure. In many cases, garbage collection is an uninteruptable procedure. If interrupt service is needed during the garbage collection, it may not be possible for the system to respond in time. Therefore, if a LISP-based expert system is to be developed, the details of how garbage collection is handled must be known to the designer, to insure that it will not interfere with the real-time functions. In addition Lisp executes slowly on a non-Lisp machine.

Development Systems

When comparing the capabilities of the commercially available expert system development packages, one of the first items to examine is the command language. The command language used should allow the user to write in the implementation language, if necessary.

Using LISP hardware and software, allows development of an expert system that will run efficiently in a real-time environment. However, there are rare exceptions, the packages can not be downloaded to any other target computer. Work is going on to improve LISP and PROLOG compilers and interpreters for standard computers.

In terms of cost, LISP hardware and software is quite high compared to a conventional computer. This situation will change as soon as low cost LISP machines become widely available. Other considerations for selecting a development system for an expert system that will be used in an industrial or process control environment include:

1. Most systems will require a representation that is hybrid in nature. The language used in the system builder should allow both rules and frames.
2. The system should be able to communicate with other computers, or with other tasks in the host computer. This allows the expert systems to communicate with each other and with conventional software.
3. Development and run-time portions of the expert system should be separable, so that run-only delivery systems can be developed.
4. Multiple inference engines should be available. This will permit the knowledge base to be designed to fit the application, not the other way around.
5. Forward chaining and backward chaining are both important in control applications.

Cognitive Overload and Operatorless Systems: It is clear that as these systems progress that more strategic functions will be incorporated into the software. The future of operatorless systems becomes a cause for wide concern. The impact is not only in union negotiations, but in productivity, competitiveness and perhaps industrial survival. The issue is one of broad social concern. Indeed, machine intelligence is a hidden time bomb in our social fabric, which may have to be dealt with long before some of the other impending societal dooms we are alledgedly facing.

Clearly, the cognitive overload problem must be faced. The level of complexity of systems is now approaching the limits of human cognitive abilities, in terms of response times and reliability. The machine as 'decision maker' or 'advisor' becomes a issue which goes beyond the normal demands placed on the designer.

Uncertain Data: Various systems builders can operate with both uncertain input data and with uncertainty associated with the consequences of the rules. Despite this facility, there is still disagreement about how the uncertainty in the physical world is best represented in the inference engine, and how is is best characterized in the input data. This field of research will continue for some time into the future.

Strategy and Heuristics: There is a need for a representation of high level strategic goals into the execution of the knowledge base. Indeed, distinguishing the point where strategy becomes tactics is of importance. There is a requirement here for process descriptions that extend over more than one snapshot of a systems state; i.e., for extended Markov models. Incorporating longer term trends into a strategic response plan is difficult and will undoubtedly depend on heuristics derived from good strategist, at least in the beginning. The study of the representation of strategy is an interesting and important area of research.

Performance: Predicting the performance of an expert system seems at this time to be closer to an experimental art rather than a reasonably theoretically based science.

The characteristics of a knowledge base are often fragile when subjected to compression or compaction, therefore, seeking speed-ups based on this approach requires great skill. The lack of an appropriate approach can lead to not finding the best solution or not finding any solution.

Results from a knowledge base are dependent on the boundary status of the rules. Interesting event combinations could cause irrational problems depending on the rule structure.

Commercial Development: It is difficult to ignore the lack of commercial results in this field. The technology has gone commercial too fast. The field is too young for this to happen. The consequences will be the duplication of results and an increased cost in the utilization of this new technology. There is considerable agruement for the research to remain in the Universities for a while longer until the baic problems have been worked out.

The Final Word

For those of us driving the technology, the challenge is to think in ever increasing levels of abstaction about our machines and sytems. We are, at this point in time, limited by concepts, not technology.

REFERENCES:

1) P. Jervis, "Standards for Real-Time Databases", Proceedings of the European Seminar on Industrial Software Engineering and the European Workshop on Industrial Computer Systems, Freiburg, Germany, Apr. 1984.

2) S. Henkind, L. Teichholz, and M. Harrison, "Intensive Care Unit Monitoring using a Real-Time Expert System", Computers in Cardiology, IEEE, Salt Lake City, UT., USA., Sep. 1984.

3) C.G. Knickerbocker, R.L. Moore, and L.B. Hawkinson, " Expert Systems Applications in Industry", Proceedings of the I.S.A. Intenational Conference and Exhibit on advances in instrumentation, Vol. 39, Houston TX, USA, Oct. 1984.

4) C.G. Knickerbocker, R.L. Moore, L.B. Hawkinson, and L.M. Churchman, "A Real-Time Expert System for Process Control", Proceeding of the first Conference on Artificial Intelligence Applications, IEEE, Denver CO., USA, Dec. 1984.

5) N.K. Gupta, and R.E. Seviora, "An Expert System Approach to Real-Time System Debugging", Proceedings of the First Conference on Artificial Intelligence Applications, IEEE, Denver CO., USA, Dec. 1984.

6) S.C. Sood, "Expert Knowledge Based Real Time Inspection System", Proceedings of the 7th International Conference on Automated Inspection and Product Control, Quality Today Magazine, Birmingham, England, Mar. 1985.

7) R.L. Moore, "Adding Real-Time Expert System Capabilities to Large Distributed Control Systems", Control Engineering, USA, Vol. 32, No. 4, Apr. 1985.

8) T.E. Murphy, "Setting up an Expert System", I&CS-Industrial and Process Control Magazine, USA, Vol. 58, No. 3, Mar. 1985.

9) C.G. Knickerbocker, R.L. Moore, L.B. Hawkinson, and M.E. Levin, "The PICON Expert System for Process Control", Fifth International Workshop on Expert Systems and their Applications, Avignon France, May 1985.

10) M.E. Sorrells, "A time-Constrained Inference Strategy for Real-Time Expert Systems", Proceedings of the IEEE National Aerospace and Electronics Conference, Dayton, OH., USA, May 1985.

11) R. Shaw, "The Potential of Expert Systems", Industrial Digital Control Systems, I.E.E. Vacation School, Oxford, England, Sep. 1985.

12) K.A. Delic, and S.U. Tripkovic, "A Relational Database Support for Real-Time Advisory Systems", Proceedings of International Conference on Industrial and Electronics, Control and Instrumentation, San Francisco, CA, USA, Nov. 1985.

13) S. Oh, "A Distributed Associative Memory for Power System Security Assessment", Proceedings of International Conference on Cybernetics and Society, Tucson, AZ, USA, Nov. 1985.

14) P. Norman, and S. Naveed, "An Expert System Supervisor for a Rotary Cement Kiln", IEE Colloquium on 'Real-Time Expert Systems in Process Control', Salford, England, Nov. 1985.

15) R.R. Leitch, " Qualitative Models of Physical Processes", IEE Colloquium on 'Real-Time Expert Systems in Process Control', Salford, England, Nov. 1985.

16) J. Lumley, "Deep Knowledge and Simulation in Knowledge-Based Control", IEE Colloquium on 'Real-Time Expert Systems in Process Control', Salford, England, Nov. 1985.

17) C. McGreavy, "Expert Systems in Proces Plant Start-up", IEE Colloquium on 'Real-Time Expert Systems in Process Control', Salford, England, Nov. 1985.

18) J.H. Efstathiou, "Expert Systems for Maintenance Tasks", IEE Colloquium on 'Real-Time Expert Systems in Process Control', Salford, England, Nov. 1985.

19) J.C. Taunton, "A Rule Based Supervisory Control System", IEE Colloquium on 'Real-Time Expert Systems in Process Control', Salford, England, Nov. 1985.

20) D.N. Shorter, "The Alvey Rescu Project - A Progress Report", IEE Colloquium on 'Real-Time Expert Systems in Process Control', Salford, England, Nov. 1985.

21) B.F. Wollenburg, "Feasibility Study for an Energy Management System Intelligent Alarm Processor", IEEE Power Industry Computer Application Conference, Tucson, AZ, USA, Nov. 1985.

22) R.L. Ennis, J.H. Griesmer, S.J. Hong, M. Karnaugh, J.K. Kastner, D.A. Klein, K.R. Milliken, M.I. Schor, and H.M. Van-Woerkom, "A Continuous Real-Time Expert System for Computer Operations", IBM Journal, USA, Vol. 30, No. 1, Jan. 1986.

23) M. Turner, "Real Time Experts", Systems International, England, Vol. 14, No. 1, Jan. 1986.

24) P.A. Sachs, A.M. Paterson, M.H.M. Turner, "Escort - An Expert System for Complex Operations in Real-Time", Expert Systems, England, Vol 3, No. 1, Jan. 1986.

25) S.L. Hardt, and J. Rosenberg, "Developing an Expert Ship Message Interpreter: Theoretical and Practical Conclusions", Optical Engineering, USA, Vol 25, No. 3, Mar. 1986.

26) M. L. Wright, M.W. Green, G. Fiegl, and P.F. Cross, "An Expert System for Real-Time Control", IEEE Software, USA, Vol. 3, No. 2, Mar. 1986.

27) B. Wollenburg, "Feasibility Study for an Energy Management System Intelligent Alarm Processor", IEEE Transactions on Power Systems, USA, Vol PWRS-1, No. 2, May 1986.

28) M.E. Ulug, "A Real-Time AI System For Military Communications", Proceedings of the I[illegible] Phoenix Conference, Phoenix, Arizona, Feb. 1987.

ACKNOWLEDGEMENTS

The author wishes to acknowledge the work of Haleh Vafaei, currently a doctoral student at Carleton University. She did the orgiginal literature search and preliminary assessment of the systems reported here. In addition, she read the manuscript in several stages of completion and provided useful recommendations for improvements. Professor N. Dawes read the manuscript and made suggestions for improvements that are also gratefully acknowledged.

B

SELECTIVE BIBLIOGRAPHY

This bibliography with Abstracts has been prepared to support AGARD Lecture Series No. 155 by the Scientific and Technical Information Branch of the U.S. National Aeronautics and Space Administration, Washington, D.C., in consultation with the Lecture Series Director, Richard Quinlivan, General Electric Company, Binghampton, New York.

UTTL: Expert systems and simulation models; Proceedings of the Seminar, Tucson, AZ, November 18, 19, 1985   *Seminar sponsored by the University of* Arizona, NASA, and Armed Forces Communications, and Electronics Association. Tucson, AZ, University of Arizona, 1986, 272 p. No individual items are abstracted in this volume.

ABS: The seminar presents papers on modeling and simulation methodology, artificial intelligence and expert systems, environments for simulation/expert system development, and methodology for simulation/expert system development. Particular attention is given to simulation modeling concepts and their representation, modular hierarchical model specification, knowledge representation, and rule-based diagnostic expert system development. Other topics include the combination of symbolic and discrete event simulation, real time inferencing, and the management of large knowledge-based simulation projects.   86/00/00 87A18561

UTTL: Model-based reasoning for automated fault diagnosis and recovery planning in space power systems

AUTH: A/ADAMS, T. L.   PAA: A/(Advanced Decision Systems, Mountain View, CA)   IN: IECEC '86; Proceedings of the Twenty-first Intersociety Energy Conversion Engineering Conference, San Diego, CA, August 25-29, 1986. Volume 3 (A87-18026 06-20). Washington, DC, American Chemical Society, 1986, p. 1763-1767.

ABS: Attention is given to the fault diagnosis module of an electrical power system and its application to fault recovery planning. It is found that model-based reasoning and expert systems techniques can be effectively combined to solve a number of problems arising in space systems power automation. Moreover, the formulation of the problem in terms of a general constraint satisfaction mechanism with domain-dependent control also facilitates application of the technology to other space automation domains.   86/00/00 87A18126

UTTL: The PICON real-time expert system tool

AUTH: A/LEINWEBER, D.; B/CARLETON, D.   PAA: B/(LISP Machines, Inc., Los Angeles, CA)   IN: NAECON 1986; Proceedings of the National Aerospace and Electronics Conference, Dayton, OH, May 19-23, 1986. Volume 4 (A87-16726 05-01). New York, Institute of Electrical and Electronics Engineers, 1986, p. 1198-1210.

ABS: The requirements for real-time expert systems in aerospace environments are discussed and illustrated with the Process Intelligent Control (PICON) system. Emphasis is placed on the demands of expanding Space Station operations. The PICON user interface includes icons, for defining schematics for the process to be controlled. LISP-based rules are chosen from an icon menu by using a mouse to point and click or by text entry. Applications of expert systems include controlling satellite electrical power, attitude control, station keeping and propulsion systems and space manufacturing processes.   86/00/00 87A16842

UTTL: The application of psychological scaling techniques in modeling expert knowledge

AUTH: A/MANHEIMER, J. M.; B/KANARSKI, T. A.   PAA: B/(Lockheed Missiles and Space Co., Inc., Austin, TX) IN: NAECON 1986; Proceedings of the National Aerospace and Electronics Conference, Dayton, OH, May 19-23, 1986. Volume 3 (A87-16726 05-01). New York, Institute of Electrical and Electronics Engineers, 1986, p. 935-940.

ABS: Developing models of expert knowledge is a goal of many projects in human factors and artificial intelligence. Unfortunately, conventional methods of knowledge elicitation such as interviewing can be time-consuming and oftentimes inaccurate. This paper identifies psychological techniques that can potentially facilitate the process of knowledge modeling. In particular, multidimensional scaling and additive similarity trees are examined as techniques that can be applied to develop models of expert knowledge in a wide variety of domains. These techniques analyze judgments of psychological distance between items in a domain. In the paper, discussions focus on the psychological rationale underlying the use of multidimensional scaling and additive similarity trees and the results of applying them in ongoing user interface and expert system development projects.   86/00/00 87A16820

UTTL: Human factors research and development requirements for future aerospace cockpit systems

AUTH: A/SCHIFFLER, R. J.; B/PINKUS, A. R.   PAA: B/(USAF, Aeronautical Systems Div., Wright-Patterson AFB, OH) IN: NAECON 1986; Proceedings of the National Aerospace and Electronics Conference, Dayton, OH, May 19-23, 1986. Volume 3 (A87-16726 05-01). New York, Institute of Electrical and Electronics Engineers, 1986, p. 883-885.

ABS: Design requirements and technologies which will heavily influence human factors R&D in cockpit design in the 1990s are discussed. Trends towards integrated displays and controls, increased use of glass or solid-state displays, night vision goggles, laser/nuclear flashblindness protection, and helmet-mounted displays are noted. Pilots will use touch-screens, voice-activated systems and chord keyboards, and expert systems and AI-generated assistance and display control. Man-in-the-loop tests are required before finalizing integrated hardware/software designs. 86/00/00 87A16813

UTTL: A new meaning to 'flying the desk'
AUTH: A/SEXTON, G. A. PAA: A/(Lockheed-Georgia Co., Marietta, GA) CORP: Lockheed-Georgia Co., Marietta. IN: NAECON 1986; Proceedings of the National Aerospace and Electronics Conference, Dayton, OH, May 19-23, 1986. Volume 2 (A87-16726 05-01). New York, Institute of Electrical and Electronics Engineers, 1986, p. 360-366.
ABS: A unique advanced transport flight station design is described. The various systems and displays of the design are described, including: the configuration; switches; tailored logic/artificial intelligence; primary flight controllers; front panel displays; primary flight/navigation display; engine power/status, approach charts, and weather display; the Advisory, Caution, and Warning System/Cockpit Display of Traffic Information display; checklist/functional systems display; head-up display; voice command and response system; Flight Management Computer system; and integrated communications/navigation system. The application of the flight station to military research is briefly discussed. 86/00/00 87A16762

UTTL: INKA - The INglish Knowledge Acquisition interface for electronic instrument troubleshooting systems
AUTH: A/PHILLIPS, B.; B/MESSICK, S. L.; C/FREILING, M. J.; D/ALEXANDER, J. H. PAA: D/(Tektronix Computer Research Laboratory, Beaverton, OR) IN: The engineering of knowledge-based systems; Proceedings of the Second Conference on Artificial Intelligence Applications, Miami Beach, FL, December 11-13, 1985 (A87-16676 05-63). Washington, DC, IEEE Computer Society Press, 1985, p. 676-681.
ABS: INKA is a natural language interface to facilitate knowledge acquisition during Expert System development for electronic instrument troubleshooting. It constrains users to create statements within a subset of English. Incremental parsing allows immediate remedial information to be generated if a user deviates from the sublanguage. Sentences are translated into production rules using the methodology of Lexical-Functional Grammar. The system is written in Smalltalk and produces rules for a Prolog inference engine. 85/00/00 87A16725

UTTL: Using rule induction to combine declarative and procedural knowledge representations
AUTH: A/RIESE, C. E.; B/ZUBRICK, S. M. PAA: B/(Radian Corp., Austin, TX) IN: The engineering of knowledge-based systems; Proceedings of the Second Conference on Artificial Intelligence Applications, Miami Beach, FL, December 11-13, 1985 (A87-16676 05-63). Washington, DC, IEEE Computer Society Press, 1985, p. 603-606.
ABS: Knowledge representation in expert systems can be viewed as having various degrees of declarative and procedural knowledge content. Large practical applications need both types of knowledge input. During the initial stages of solving a hard problem, most knowledge is entered in declarative form. As the application matures, procedural knowledge is added to obtain a more direct and efficient traversal of the problem search space. Rule induction is a technique for automatically generating efficient procedural algorithms from declarative knowledge (in the form of examples). Hierarchical organization of knowledge acquired by inductive learning is the basis of the RuleMaster expert system building package. Inductive learning of procedures from examples was used to build a severe storm forecasting expert system. 85/00/00 87A16719

UTTL: A retrospective view of CACE-III - Considerations in coordinating symbolic and numeric computation in a rule-based expert system
AUTH: A/JAMES, J. R.; B/FREDERICK, D. K.; C/BONISSONE, P. P.; D/TAYLOR, J. H. PAA: A/(U.S. Military Academy, West Point, NY); B/(Rensselaer Polytechnic Institute, Troy, NY); D/(General Electric Co., Schenectady, NY) IN: The engineering of knowledge-based systems; Proceedings of the Second Conference on Artificial Intelligence Applications, Miami Beach, FL, December 11-13, 1985 (A87-16676 05-63). Washington, DC, IEEE Computer Society Press, 1985, p. 532-538.
ABS: The experience derived from designing and implementing CACE-III (Computer-Aided Control Engineering third-generation environment) is described. CACE-III is a knowledge-based system that requires the coordination of symbolic and numeric computation. Its purpose is to aid a user in applying available analysis and synthesis software to complete the steps in the control system engineering process. Help is provided in modeling, constraining, diagnosing, specifying, designing and simulating dynamical

systems. The description of CACE-III will cover the following topics: partitioning the knowledge corresponding to different stages in the design process, implementing a lead-lag compensator design heuristic, developing a limited form of nonmonotonic reasoning to allow for iterations and trade-offs during the design process, translating numerical data into symbolic form, generating a sequence of commands for the numerical routines, and coordinating the numerical computation of the routines with the symbolic computation of the inference engine.
85/00/00 87A16713

UTTL: Progress in knowledge-based flight monitoring
AUTH: A/CHEN, D. C. PAA: A/(E-Systems, Inc., Garland Div., Dallas, TX) CORP: E-Systems, Inc., Dallas, Tex. IN: The engineering of knowledge-based systems; Proceedings of the Second Conference on Artificial Intelligence Applications, Miami Beach, FL, December 11-13, 1985 (A87-16676 05-63). Washington, DC, IEEE Computer Society Press, 1985, p. 441-446.
ABS: Features and applications of the script-based flight monitor SECURE are described. Implemented on an on-board computer, SECURE treats a flight as a regular sequence of contexts (situations) defined in a knowledge base with a hierarchical structure for successively more finely delineated flight phases, i.e., takeoff, cruise and landing. SECURE provides normalcy references for flight monitoring and allows context identification, which allows the presentation of checklists. An implementation of SECURE, written in MACLISP, on a DC-10 flight simulator is described.
85/00/00 87A16705

UTTL: Generic tasks in knowledge-based reasoning - Characterizing and designing expert systems at the 'right' level of abstraction
AUTH: A/CHANDRASEKARAN, B. PAA: A/(Ohio State University, Columbus) IN: The engineering of knowledge-based systems; Proceedings of the Second Conference on Artificial Intelligence Applications, Miami Beach, FL, December 11-13, 1985 (A87-16676 05-63). Washington, DC, IEEE Computer Society Press, 1985, p. 294-300.
ABS: A generalized approach to the development of expert systems is presented. The approach is based on the view that complex knowledge-based reasoning tasks can be decomposed into generic tasks. Each task is associated with specific types of knowledge and control regimes. Six commonly applied generic tasks, i.e., classification, state abstraction, knowledge-directed retrieval, object synthesis by plan selection and refinement, hypothesis matching, and assembly of compound hypotheses for abduction, are described for use in constructing expert systems. Applications such as medical diagnosis, design tasks, and abstract reasoning are discussed. 85/00/00 87A16699

UTTL: Incorporating knowledge rules in a semantic data model - An approach to integrated knowledge management
AUTH: A/SU, S. Y. W.; B/RASCHID, L. PAA: B/(Florida, University, Gainesville) IN: The engineering of knowledge-based systems; Proceedings of the Second Conference on Artificial Intelligence Applications, Miami Beach, FL, December 11-13, 1985 (A87-16676 05-63). Washington, DC, IEEE Computer Society Press, 1985, p. 250-256.
ABS: This paper presents a framework for an integrated knowledge base management system. The integration of a rule base of knowledge rules with a database of facts or assertions is a key concept in this KBMS and provides for more efficient knowledge processing and management. The objects of the knowledge base comprise facts and relevant rules; this is supported by a technique of incorporating knowledge rules in a semantic data model. The KBMS supports complete specification of the declarative and operational semantics of various kinds of knowledge rules including integrity and security constraints, deductive rules and expert rules. A mechanism for the automatic triggering of knowledge rules and for incorporating these rules into a sequence of operations in a transaction is supported by the KBMS.
85/00/00 87A16697

UTTL: The role of AI in computer vision
AUTH: A/SHAPIRO, L. G. PAA: A/(Machine Vision International, Ann Arbor, MI) IN: The engineering of knowledge-based systems; Proceedings of the Second Conference on Artificial Intelligence Applications, Miami Beach, FL, December 11-13, 1985 (A87-16676 05-63). Washington, DC, IEEE Computer Society Press, 1985, p. 76-81.
ABS: The requirements of computer vision systems (CVS) are discussed, along with features of current CVS and the roles of AI in present and future CVS. The interactions among low-, middle- and high-level vision systems are reviewed, noting that only high-level systems (HLS) will be guided by AI. HLS involves matching and reasoning tasks and the coordination of mid- and low-level processes. Features of the experimental VISIONS, ACRONYM and SPAM CVSs are described, with emphasis on techniques used to teach and control the systems. 85/00/00 87A16683

UTTL Sensor-based fault diagnosis in a flight expert system
AUTH A/ALI, M.; B/SCHARNHORST, D. A. PAA: B/(Tennessee, University, Tullahoma) CORP: Tennessee Univ., Tullahoma. IN: The engineering of knowledge-based systems; Proceedings of the Second Conference on Artificial Intelligence Applications, Miami Beach, FL, December 11-13, 1985 (A87-16676 05-63). Washington, DC, IEEE Computer Society Press, 1985, p. 49-54. FAA-supported research.
ABS: A prototype of a knowledge-based flight expert system (FLES) has been developed to assist airplane pilots in monitoring, analyzing, and diagnosing faults and to provide support in reducing the pilot's own mistakes. A sensor simulation model has been developed to provide FLES with the airplane status information during the diagnostic process. The simulator is based partly on the Advanced Concept System (ACS), a future-generation airplane, and partly on the Boeing 737, an existing airplane. The architecture of FLES contains several subsystems. One of the major subsystems performs fault diagnosis in the electrical system of the ACS. This paper describes the mechanism and functionality of the automatic diagnosis performed in this expert system. 85/00/00 87A16681

UTTL: Evolution and modification of probability in knowledge bases
AUTH: A/WINTER, C. L.; B/GIRSE, R. D. PAA: A/(Science Applications, Inc., Tucson, AZ); B/(Idaho State University, Pocatello, ID) IN: The engineering of knowledge-based systems; Proceedings of the Second Conference on Artificial Intelligence Applications, Miami Beach, FL, December 11-13, 1985 (A87-16676 05-63). Washington, DC, IEEE Computer Society Press, 1985, p. 27-31.
ABS: The assignment and modification of certainty in the development of knowledge bases for expert systems is discussed from a probabilistic point of view. The Law of Large Numbers (LLN) is examined and, in contrast to Bayes statistics, is demonstrated to lead to true certainties by attaching probabilities to an entire rule instead of just to the edges of rules. Further, it is shown that inference chains which retain a high probability can be treated as rules to establish learning through efficiency, thus generating rules not originally included in the knowledge base. 85/00/00 87A16678

UTTL: The engineering of knowledge-based systems; Proceedings of the Second Conference on Artificial Intelligence Applications, Miami Beach, FL, December 11-13, 1985 Conference sponsored by IEEE. Washington, DC, IEEE Computer Society Press, 1985, 695 p. For individual items see A87-16677 to A87-16725.
ABS: The engineering of knowledge-based systems is examined with emphasis on the development and applications of AI. Methods for performing reasoning functions in the presence of large uncertainties, for diagnostic evaluations and for electronic instrument and flight applications are discussed. The use of AI for enhancing the performance of computer vision systems, mission planning for a mobile robot and for opportunistic scheduling for robotic machine tending is explored. Expert systems architectures are described, including a fuzzy inference engine, a behavior rule-based system for distributed process control, and a system for rule-based analysis of programming environments. 85/00/00 87A16676

UTTL: The concept of autonomous flight management system for future spacecraft
AUTH: A/TANABE, T.; B/HARIGAE, M.; C/TOMITA, N. PAA: C/(Tokyo, University, Japan) IAF, International Astronautical Congress, 37th, Innsbruck, Austria, Oct. 4-11, 1986. 9 p.
ABS: The concept of an autonomous flight management system for future spacecraft based on artificial intelligence (AI) technology is discussed. AI techniques are applied in the sense that parametric relationships between relevant system parameters can be learned during the operation or a special learning process. The results are stored in a knowledge data base which is used again during the operation or the learning process. The discussion covers the scope of an autonomous flight management system, important aspects of a knowledge data base, a self-learning system for the knowledge data base, and simulated results.
RPT#: IAF PAPER 86-01 86/10/00 87A15801

UTTL: Expert system architecture for control system design
AUTH A/TRANKLE, T. L.; B/SHEU, P.; C/RABIN, U. H. PAA: C/(Systems Control Technology, Inc., Palo Alto, CA) IN: 1986 American Control Conference, 5th, Seattle, WA, June 18-20, 1986, Proceedings. Volume 2 (A87-13301 03-63). New York, Institute of Electrical and Electronics Engineers, 1986, p. 1163-1169. Research supported by McDonnell Aircraft Corp
ABS: This paper describes the development of an expert planning framework for design of control systems. The

expert planner incorporates the knowledge of a control system designer in selecting and applying mathematical algorithms from several branches of control theory such as Kalman filter design, proportional/integral/derivative control, and optimal state feedback. The framework has been developed directly in Lisp rather than using any of the available domain-independent expert system frameworks. The planning is done at two levels: a high level that uses general rules about control system design and a low level that uses rules about the use of a computer aided control system design (CACSD) package. The system has been implemented and demonstrated on a Xerox 1100 Lisp workstation controlling a CACSD packagte running on a VAX 11/780, designing a servo (command tracking) control system for aircraft. 86/00/00 87A13410

UTTL: An expert system using computer algebra for design of nonlinear control systems
AUTH: A/SU, L. S.; B/BLANKENSHIP, G. L. PAA: B/(Maryland, University, College Park) IN: 1986 American Control Conference, 5th, Seattle, WA, June 18-20, 1986, Proceedings. Volume 2 (A87-13301 03-63). New York, Institute of Electrical and Electronics Engineers, 1986, p. 1151, 1152.
ABS: A prototype expert system for the treatment of stochastic control- and nonlinear signal processing problems has been developed in a collaborative effort with INRIA. The system is written for the most part in MACSYMA and LISP. It accepts user input (models and design objectives) in symbolic form; it carries out the basic analysis of the user's problem in symbolic form using MACSYMA; and it produces output in the form of automatically generated FORTRAN programs for the final numerical reduction of the problem. A component of the system which supports the design analysis of nonlinear control systems is described. The methodology is based on the Hunt Su-Meyer technique for exact or approximate linearization of nonlinear control systems using coordinate transformations based on the intrinsic differential geometric structure of the control system, i.e., its controllability properties and Kronecker indices. Sample sessions are presented to illustrate operation of the software. The status of the overall expert system and plans for its further development are described. 86/00/00 87A13409

UTTL: Time-based air traffic management using expert systems
AUTH: A/TOBIAS, L.; B/SCOGGINS, J. L. PAA: B/(NASA, Ames Research Center, Moffett Field, CA) CORP: National Aeronautics and Space Administration. Ames Research Center, Moffett Field, Calif. IN: 1986 American Control Conference, 5th, Seattle, WA, June 18-20, 1986, Proceedings. Volume 2 (A87-13301 03-63). New York, Institute of Electrical and Electronics Engineers, 1986, p. 693-700.
ABS: A prototype expert system has been developed for the time scheduling of aircraft into the terminal area. The three functions of the air-traffic-control schedule advisor are as follows: (1) for each new arrival, it develops an admisible flight plan for that aircraft; (2) as the aircraft progresses through the terminal area, it monitors deviations from the aircraft's flight plan and provides advisories to return the aircraft to its assigned schedule; and (3) if major disruptions such as missed approaches occur, it develops a revised plan. The advisor is operational on a Symbolics 3600, and is programmed in MRS (a logic programming language), Lisp, and Fortran. 86/00/00 87A13362

UTTL: Generic tasks in knowledge-based reasoning - High-level building blocks for expert system design
AUTH: A/CHANDRASEKARAN, B. PAA: A/(Ohio State University, Columbus) IEEE Expert (ISSN 0885-9000), vol. 1, Fall 1986, p. 23-30. DARPA-supported research.
ABS: The use of generic tasks in expert system is examined. The development of knowledge representation and control regimes for diagnostic reasoning and design problems is discussed. The six generic tasks, hierarchical classification, hypothesis matching or assessment, knowledge-directed information passing, abductive assembly, hierarchical design by plan section and refinement, and state abstraction, used to construct knowledge-based systems are described. Existing expert systems are analyzed in terms of these generic tasks and complex generic tasks such as process-control problems are considered. The creation of a correlation between the form of knowledge and its applications is studied. 86/00/00 87A12234

UTTL: Monitoring real-time navigation processes using the automated reasoning tool (ART)
AUTH: A/MALETZ, M. C.; B/CULBERT, C. J. PAA: A/(Inference Corp., Los Angeles, CA); B/(NASA, Johnson Space Center, Houston, TX) CORP: National Aeronautics and Space Administration. Lyndon B. Johnson Space Center, Houston, Tex. IN: Annual Aerospace Applications of

Artificial Intelligence Conference, 1st, Dayton, OH, September 16-19, 1985, Proceedings (A87-12211 02-63). Dayton, OH, AAAIC Secretariat, 1985, p. 350-360.
ABS: An expert system is described for monitoring and controlling navigation processes in real-time. The ART-based system features data-driven computation, accommodation of synchronous and asynchronous data, temporal modeling for individual time intervals and chains of time intervals, and hypothetical reasoning capabilities that consider alternative interpretations of the state of navigation processes. The concept is illustrated in terms of the NAVEX system for monitoring and controlling the high speed ground navigation console for Mission Control at Johnson Space Center. The reasoning processes are outlined, including techniques used to consider alternative data interpretations. Installation of the system has permitted using a single operator, instead of three, to monitor the ascent and entry phases of a Shuttle mission. 85/00/00 87A12220

UTTL: Technical diagnosis by automated reasoning
AUTH: A/ALLEN, D.; B/RADER, K. PAA: B/(Northrop Corp., Aircraft Div., Hawthorne, CA) IN: Annual Aerospace Applications of Artificial Intelligence Conference, 1st, Dayton, OH, September 16-19, 1985, Proceedings (A87-12211 02-63). Dayton, OH, AAAIC Secretariat, 1985, p. 185-194.
ABS: Features and applications of an automated diagnostic program (IN-ATE/2) which includes a rule-based heuristic search strategy for aiding a technician trouble-shooting a system element are described. The element is represented as a logic model, which permits the system to generate its own rule base. Introduction of a true rulebase, i.e., expert knowledge, changes the program to a true expert system. The logic model consists of a topological model of the system being examined. Output is a recommended optimum sequence of tests based on a binary decision tree. The goal of the output is to isolate the fault as quickly and cheaply as possible, and to identify the point at which tests must be made with individual components. A sample application of IN-ATE/2 to trouble-shoot a problem in the circuitry of an F-5 aircraft is outlined. Planned enhancements and implementations of the IN-ATE/2 are discussed. 85/00/00 87A12217

UTTL: Expert systems for aiding combat pilots
AUTH: A/ANDERSON, B. M.; B/MCNULTY, C.; C/LYSTAD, G. S. PAA: C/(Texas Instruments, Inc., Dallas) IN: Annual Aerospace Applications of Artificial Intelligence Conference, 1st, Dayton, OH, September 16-19, 1985, Proceedings (A87-12211 02-63). Dayton, OH, AAAIC Secretariat, 1985, p. 54-60.
ABS: The emergency procedures expert system (EPES) for handling emergency situations for tactical aircraft pilots is described. EPES is activated when the aircraft is damaged and loses capabilities, thereby initiating mission replanning. The system must provide advice as to whether or not to pursue a particular course of action, and must resolve conflicts between goals. The latter arise when opposing actions are specified by multiple emergencies. Performance of the system is illustrated by means of sample F-16 missions. In one scenario, the EPES must provide advice on correct actions when the canopy is lost and the towershaft fails, situations which require recommended airspeeds of a maximum of 180 and a minimum of 250 kt, respectively. The prosvision of mission options for sequential failures on a deepstrike mission is also described, and a block diagram is provided of the multiple expert system architecture. 85/00/00 87A12212

UTTL: Annual Aerospace Applications of Artificial Intelligence Conference, 1st, Dayton, OH, September 16-19, 1985, Proceedings
AUTH: A/JOHNSON, J. R. PAA: A/(USAF, Wright Aeronautical Laboratories, Wright-Patterson AFB, OH) Conference sponsored by NCR, VEDA, Inc., Boeing Military Airplane Co., et al. Dayton, OH, AAAIC Secretariat, 1985, 359 p. For individual items see A87-12212 to A87-12220.
ABS: Topics of interest at the forefront of the development and implementation of artificial intelligence (AI) systems are detailed. Consideration is given to AI applications in the avionics displays in fighter aircraft, in manufacturing and in ground-based Shuttle navigation monitoring systems. An implementation of LISP on a SIMD is described, along with frame-based knowledge representation for processing planning. Progress in the use of expert systems in an advisory role for the design of video display units and for trouble-shooting system problems in a tactical fighter is assessed. Finally, design and performance features and aplications of the ASPRO parallel processor inference engine are summarized. 85/00/00 87A12211

UTTL: Experts system control of autonomous airborne vehicle
AUTH: A/GILMORE, J. F.; B/PEMBERTON, W. G. PAA: A/(Georgia Institute of Technology, Atlanta); B/(Martin Marietta Corp., Orlando, FL) Unmanned Systems, vol. 4, Summer 1985, p. 8-13.
ABS: Attention is given to the application of AI to airborne autonomous vehicle systems encompassing sensor vision, mission planning, and mission control tasks. Vision furnishes local and global scene analyses that are symbolically represented and passed on to planning, thereby providing initial route planning constraints. Planning then generates a task dependent path for vehicle traversal, assuring maximum system safety as well as effectiveness. Control, finally, validates the path and either executes the given route or feeds back to the previous two systems in order to resolve conflicts. A typical autonomous airborne vehicle mission in which the autonomous flight vehicle is a helicopter is presented. 85/00/00 86A49476

UTTL: An artificial intelligence approach to onboard fault monitoring and diagnosis for aircraft applications
AUTH: A/SCHUTTE, P. C.; B/ABBOTT, K. H. PAA: B/(NASA, Langley Research Center, Hampton, VA) CORP: National Aeronautics and Space Administration. Langley Research Center, Hampton, Va. AIAA, Guidance, Navigation and Control Conference, Williamsburg, VA, Aug. 18-20, 1986. 9 p.
ABS: Real-time onboard fault monitoring and diagnosis for aircraft applications, whether performed by the human pilot or by automation, presents many difficult problems. Quick response to failures may be critical, the pilot often must compensate for the failure while diagnosing it, his information about the state of the aircraft is often incomplete, and the behavior of the aircraft changes as the effect of the failure propagates through the system. A research effort was initiated to identify guidelines for automation of onboard fault monitoring and diagnosis and associated crew interfaces. The effort began by determining the flight crew's information requirements for fault monitoring and diagnosis and the various reasoning strategies they use. Based on this information, a conceptual architecture was developed for the fault monitoring and diagnosis process. This architecture represents an approach and a framework which, once incorporated with the necessary detail and knowledge, can be a fully operational fault monitoring and diagnosis system, as well as providing the basis for comparison of this approach to other fault monitoring and diagnosis concepts. The architecture encompasses all aspects of the aircraft's operation, including navigation, guidance and controls, and subsystem status. The portion of the architecture that encompasses subsystem monitoring and diagnosis was implemented for an aircraft turbofan engine to explore and demonstrate the AI concepts involved. This paper describes the architecture and the implementation for the engine subsystem.
RPT#: AIAA PAPER 86-2093 86/08/00 86A48577

UTTL: A theory for fault-tolerant flight control combining expert system and analytical redundancy concepts
AUTH: A/STENGEL, R. F.; B/HANDELMAN, D. A. PAA: A/(Princeton University, NJ) IN: Guidance, Navigation and Control Conference, Williamsburg, VA, August 18-20, 1986, Technical Papers (A86-47401 23-63). New York, American Institute of Aeronautics and Astronautics, 1986, p. 375-384.
ABS: This paper presents a theory for rule-based fault-tolerant flight control. The objective is to define methods for designing control systems capable of accommodating a wide range of aircraft failures, including sensor, control, and structural failures. A software architecture is described that integrates quantitative analytical redundancy techniques and heuristic expert system concepts for the purpose of in-flight, real-time fault tolerance. The resultant controller uses a rule-based expert system approach to transform the problem of failure accommodation task scheduling and selection into a problem of search. Control system performance under sensor and control failures is demonstrated using linear discrete-time deterministic simulations of a tandem-rotor helicopter's dynamics. It is found that the rule-based control theory can be used to enhance existing redundancy management systems. This approach to control system design also provides inherent parallelism for computational speed, smooth integration of algorithmic and heuristic computation, a search-based decision-making mechanism, straightforward system organization and debugging, and an incremental growth capability.
RPT#: AIAA PAPER 86-2092 86/00/00 86A47442

UTTL: RT-BUILD - Automatic generation of Ada code for flight control applications
AUTH: A/LEHMAN, L. L.; B/HOUTCHENS, S. P.; C/NAVAB, M.; D/SHAH, S. C. PAA: D/(Integrated Systems, Inc., Palo Alto, CA) IN: Guidance, Navigation and Control Conference, Williamsburg, VA, August 18-20, 1986.

Technical Papers (A86-47401 23-63). New York, American Institute of Aeronautics and Astronautics, 1986, p. 352-356.

ABS: Key issues in developing real-time control software for flight control applications are discussed, and the relationship of RT-BUILD to these issues is examined. RT-BUILD is an expert control system programmer that creates systems. The capabilities of RT-BUILD are reviewed, and the two major components of its implementation, the Ada source code generator and the debug executive, and the crucial architectural details of stand-along control applications that are generated by RT-BUILD are discussed. RT-BUILD application is examined by viewing it in terms of four distinct parts: a system initializer, a periodic application-level scheduler, synchronous I/O drivers, and subsystem update computations.

RPT#: AIAA PAPER 86-2088 86/00/00 86A47439

UTTL: User interaction with a control design expert system

AUTH: A/TRANKLE, T. L.; B/PEHOUSHEK, D.; C/SHEU, P. PAA: C/(Systems Control Technology, Inc., Palo Alto, CA) IN: Guidance, Navigation and Control Conference, Williamsburg, VA, August 18-20, 1986, Technical Papers (A86-47401 23-63). New York, American Institute of Aeronautics and Astronautics, 1986, p. 346-351.

ABS: This paper described an expert system having expertise in the design of linear servo command tracking control systems. The inference engine uses both backward and forward chaining rules. Backward chaining rules outline possible approaches while forward chaining rules indicate which approach has the highest probability of success. The user interacts with the system using a high resolution screen and a mouse pointing device. Windows in the screen display tree data structures describing mathematical models used in the design process and lines of reasoning about design methods, performance data on tentative designs, and fundamental mathematical operations executed during the design.

RPT#: AIAA PAPER 86-2086 86/00/00 86A47438

UTTL: Boeing developing new cockpit displays to ease pilot workload

AUTH: A/MERRIFIELD, J. T. Aviation Week and Space Technology (ISSN 0005-2175), vol. 124, June 23, 1986, p. 109, 110, 113.

ABS: A program being pursued to ready advanced cockpit displays and automated systems in order to manage the large volume of information that will be available to pilots of the USAF's advanced tactical fighter (ATF) is discussed. In the pictorial format cockpit concept being utilized, the ATF pilot will be assisted by an 'electronic copilot' equipped to automatically handle such tasks as flight and powerplant control, weapons management, and countermeasures. The system would also manage communications and navigation systems while informing the pilot of current tactical situations and advising him in the decision-making process. The research funding for this effort is discussed, and the simulation equipment utilized is described. The roles of the crew station information manager in the system is addressed, including his interaction with the pilot. A planned Kickoff demonstration of the system is described, and related developments in avionics are summarized. 86/06/23 86A45066

UTTL: An expert system for control system design

AUTH: A/TRANKLE, T. L.; B/MARKOSIAN, L. Z. PAA: B/(Systems Control Technology, Inc., Palo Alto, CA) IN: Control 85; Proceedings of the International Conference, Cambridge, England, July 9-11, 1985. Volume 2 (A86-40751 19-63). London/New York, Institution of Electrical Engineers/IEE Inspec, 1985, p. 495-499. Research supported by the McDonnell Aircraft Corp.

ABS: Work in progress to develop knowledge-based expert systems to supervise adaptive control systems in real time is reported. The overall structure of an expert adaptive controller is outlined, and a detailed description is given of one of its components: the feedback control system designer. This designer uses a planning expert system to supervise the application of computer-aided control system design (CACSD) algorithms. The design of an aircraft feedback control system is mentioned as an example. 85/00/00 86A40774

UTTL: Fifth generation computer systems 1984; Proceedings of the International Conference, Tokyo, Japan, November 6-9, 1984 Tokyo/Amsterdam and New York, OHMSHA, Ltd./North-Holland, 1984, 723 p. No individual items are abstracted in this volume.

ABS: Progress in Japan's 10-year Fifth Generation Computer Project is reported in this collection of papers. Research and development reports from the Institute for New Generation Computer Technology are presented. Additional general topics covered include: foundations for logic programs, logic programming languages/methodologies, and architectures and applications of new generation computing. 84/00/00 86A38549

UTTL: A computer controlled autopilot and an annunciation system for light aircraft
AUTH: A/DWIVEDI, S. N. PAA: A/(North Carolina, University, Charlotte) IN: Computers in engineering 1985; Proceedings of the International Computers in Engineering Conference and Exhibition, Boston, MA, August 4-8, 1985. Volume 3 (A86-36851 16-31). New York, American Society of Mechanical Engineers, 1985, p. 111-119.
ABS: A design is described for an integrated autopilot for a light aircraft, that not only flies the airplane but also monitors its systems. In the event of any malfunction, the onboard computer announces the problem and on command follows the remedial procedure. Features of the system include the use of a synthesized voice to prompt the pilot or announce malfunctions and the use of a microcomputer to monitor the aircraft systems. The system incorporates the best features of several existing autopilots and consists of nine different processors for the following functions: system controller, backup, autopilot, navigation, integrated data control, radio adapter unit, electronic horizontal situation indicator, Votrax speech units, and the hand-held computer.
85/00/00 86A36871

UTTL: An expert system for choosing the best combination of options in a general-purpose program for automated design synthesis
AUTH: A/ROGERS, J. L., JR.; B/BARTHELEMY, J.-F. PAA: A/(NASA, Langley Research Center, Hampton, VA); B/(Virginia Polytechnic Institute and State University, Blacksburg) CORP: National Aeronautics and Space Administration. Langley Research Center, Hampton, Va.; Virginia Polytechnic Inst. and State Univ., Blacksburg. IN: Computers in engineering 1985; Proceedings of the International Computers in Engineerin⁻ Co.ference and Exhibition, Boston, MA, August 4-8, 1985. Volume 2 (A86-36851 16-31). New York, American Society of Mechanical Engineers, 1985, p. 255-260.
ABS: An expert system to aid users of the Automated Design Synthesis (ADS) general-purpose optimization program has been developed. ADS has approximately 100 combinations of strategy, optimizer, and search options from which to choose. This expert system aids the user in choosing the best combination of options for solving a particular problem. The system is written in LISP, contains about 200 rules, and executes on DEC-VAX and IBM PC/XT computers.
85/00/00 86A36859

UTTL: Perspectives on knowledge in engineering design
AUTH: A/RASDORF, W. J. PAA: A/(North Carolina State University, Raleigh) CORP: North Carolina State Univ., Raleigh. IN: Computers in engineering 1985; Proceedings cf the International Computers in Engineering Conference and Exhibition, Boston, MA, August 4-8, 1985. Volume 2 (A86-36851 16-31). New York, American Society of Mechanical Engineers, 1985, p. 249-253. Research sponsored by North Carolina State University and NASA.
ABS: Various perspectives are given of the knowledge currently used in engineering design, specifically dealing with knowledge-based expert systems (KBES). Constructing an expert system often reveals inconsistencies in domain knowledge while formalizing it. The types of domain knowledge (facts, procedures, judgments, and control) differ from the classes of that knowledge (creative, innovative, and routine). The feasible tasks for expert systems can be determined based on these types and classes of knowledge. Interpretive tasks require reasoning about a task in light of the knowledge available, where generative tasks create potential solutions to be tested against constraints. Only after classifying the domain by type and level can the engineer select a knowledge-engineering tool for the domain being considered. The critical features to be weighed after classification are knowledge representation techniques, control strategies, interface requirements, compatibility with traditional systems, and economic considerations. 85/00/00 86A36858

UTTL: Cockpit automation - In need of a philosophy
AUTH: A/WIENER, E. L. PAA: A/(Miami, University, FL) CORP: Miami Univ., Fla. IN: Aerospace Behavioral Engineering Technology Conference, 4th, Long Beach, CA, October 14-17, 1985, Proceedings (A86-35426 15-54). Warrendale, PA, Society of Automotive Engineers, Inc., 1985, p. 369-375.
ABS: Concern has been expressed over the rapid development and deployment of automat c devices in transport aircraft, due mainly to the human interface and particularly the role of automation in inducing human error. The paper discusses the need for coherent philosophies of automation, and proposes several approaches: (1) flight management by exception, which states that as long as a crew stays within the bounds of regulations, air traffic control and flight safety, it may fly as it sees fit; (2) exceptions by forecasting, where the use of forecasting models would predict boundary penetration, rather than waiting for it to happen; (3) goal-sharing, where a computer is informed of overall goals, and subsequently has the

capability of checking inputs and aircraft position for consistency with the overall goal or intentions; and (4) artificial intelligence and expert systems, where intelligent machines could mimic human reason.
RPT#: SAE PAPER 851956 85/00/00 86A35454

UTTL: Applications of knowledge-based systems to engineering analysis and design; Proceedings of the Winter Annual Meeting, Miami Beach, FL, November, 17-22, 1985
AUTH: A/DYM, C. L. PAA: A/(Massachusetts, University, Amherst) Meeting sponsored by ASME. New York, American Society of Mechanical Engineers (Aerospace Symposia Series. Volume AD-10), 1985, 153 p. For individual items see A86-34402 to A86-34406.
ABS: The present conference presents papers on expert systems for mechanical design, knowledge-based analysis and design systems for aerospace structures, and intelligent gateway processors as integrators of CAD/CAM networks. Also considered are a framework for a knowledge-based finite element analysis assistant, an environment for building engineering knowledge-based systems, and a knowledge-based system for updating engineering project schedules. Other topics include the PRIDE expert system for the design of paper handling systems, the SITECHAR expert system component of a geotechnical side characterization workbench, and an application of expert systems to composite structural design and analysis. 85/00/00 86A34401

UTTL: A user preference guided approach to conflict resolution in rule-based expert systems
AUTH: A/WHITE, C. C., III; B/SYKES, E. A. PAA: B/(Virginia, University, Charlottesville) IEEE Transactions on Systems, Man, and Cybernetics (ISSN 0018-9472), vol. SMC-16, Mar.-Apr. 1986, p. 276-278. DCA-supported research.
ABS: An approach to conflict resolution in rule-based expert systems that incorporates a behaviorally relevant generalization of multiattribute utility theory is proposed. The intent of this approach is to take into account preferential knowledge, specific to the user and/or situation, in rule selection. This knowledge will enhance the user's perception of the system's capability and hence the system's acceptability. 86/04/00 86A34164

UTTL: Artificial intelligence research at the APL research center An overview
AUTH: A/SIGILLITO, V. G. PAA: A/(Johns Hopkins University, Laurel, MD) Johns Hopkins APL Technical Digest (ISSN 0270-5214), vol. 7, Jan.-Mar. 1986, p. 15-18.
ABS: Artificial intelligence (AI), defined as the computer science subfield that uses specific domain knowledge and heuristics to solve exponentially hard problems, is discussed and some current AI projects are described. Nonalgorithmic AI programs emphasize symbolic computations, and are concerned with learning, knowledge representation, and reasoning. Reasons for AI research include the diminishing time available to make decisions which are based on an exponentially increasing body of knowldge. Projects are discussed on the development of a general machine vision system, and on the acquisition and representation of knowledge for distributed command decision assistance. Other projects described are a computer environment for automating software development, and an environment for the design, implementation and evaluation of languages for knowledge represenation and acquisition. 86/03/00 86A33831

UTTL: The use of signal detection theory in research on human-computer interaction
AUTH: A/PARASURAMAN, R.; B/WISDOM, G. PAA: B/(Catholic University of America, Washington, DC) IN: Human Factors Society, Annual Meeting, 29th, Baltimore, MD, September 29-October 3, 1985, Proceedings. Volume 1 (A86-33776 15-53). Santa Monica, CA, Human Factors Society, 1985, p. 33-37.
ABS: The use of signal detection theory (SDT) in evaluating attentional and decision-making performance in the context of human-computer interaction is reviewed. SDT provides a means for distinguishing between accuracy and criterion setting in decision-making environments. This is useful in evaluating the effectiveness of the decision-making performance of an intelligent machine, a human user, or a human-machine system. SDT is also useful in deciding how best to allocate subtasks and functions in human-computer monitoring systems. Applications of SDT to the evaluation of rule bases in expert systems and the design of computer assistance in human-computer monitoring are discussed. 85/00/00 86A33779

UTTL: Expert control
AUTH: A/ASTROM, K. J.; B/ANTON, J. J. PAA: A/(Lund Institute of Technology, Sweden); B/(Systems Control Technology, Inc., Palo Alto, CA) IN: A bridge between control science and technology. Volume 5 (A86-33181 14-63). Oxford and New York, Pergamon Press, 1985, p. 2579-2584.
ABS: There has been substantial progress in theory and practice of automatic control through application of mathematical analysis and numerics. Numerical data processing has, however, so far only had marginal influence on control systems. Actual implementations of control systems also contain a substantial amount of heuristic logic. The paper shows that this logic may be replaced by an expert system. This leads to simplifications in implementation as well as new capabilities in the control system. 85/00/00 86A33185

UTTL: Cockpits for 2010 and beyond
AUTH: A/SUMMERS, P. I. PAA: A/(McDonnell Aircraft Co., St Louis, MO) IEEE Aerospace and Electronic Systems Magazine (ISSN 0885-8985), vol. 1, Feb. 1986, p. 17-20.
ABS: A proposed cockpit design for fighter aircraft is described. The ground-based mission planning system is discussed. The use of a forebody concept to accommodate the physiological needs of the crew is examined. An armrest control module is proposed for controlling attitude and thrust. The encapsulation of the crew station and display systems such as the helmet mounted sight and display system are studied. The application of artificial intelligence and biocybernetics to the cockpit of the aircraft is analyzed. 86/02/00 86A31849

UTTL: Expert systems in the fighter of the 1990s
AUTH: A/YANNONE, R. M. PAA: A/(General Electric Co., Aerospace Electronic Systems Dept., Utica, NY) IEEE Aerospace and Electronic Systems Magazine (ISSN 0885-8985), vol. 1, Feb. 1986, p. 12-16.
ABS: The development of a multibeam system for the Advanced Tactical Fighter based on mission requirements is examined. The multisensor data fusion and situation assessment expert systems are described. The fusion sensor operates over a region of the surveillance volume asynchronously and provides data to maximize target identification and kinematic state vector accuracy. The measurement set level, track file level, and onboard common aperture arrangements for the fusion system are discussed. A block diagram for processing the autonomous sensor track files is provided. The situation assessment control sensor utilizes real-time sensor and historical data to provide display parameters for the pilots, weapon cueing, countermeasure response management, and data collection direction. A table listing the subprocesses of the two expert systems is presented. 86/02/00 86A31848

UTTL: Knowledge-based approach toward developing an autonomous helicopter system
AUTH: A/GILMORE, J. F.; B/SEMECO, A. C. PAA: B/(Georgia Institute of Technology, Atlanta) Optical Engineering (ISSN 0091-3286), vol. 25, March 1986, p. 415-427. Research supported by the Georgia Institute of Technology.
ABS: This paper presents a description of the Autonomous Helicopter System (AHS) which is currently being implemented in the Georgia Tech Research Institute. It is pointed out that autonomous vehicles provide a mechanism for removing humans from modern-day battlefields while not impacting the tactical capabilities of the battle force. Civilian applications for such a device include the investigation of hazardous areas, search over large areas, and the monitoring of large facilities. The AHS consists of three sections. The vision section is the sensor processing component of the AHS. Attention is given to image segmentation, region classification, scene analysis, visual model construction, scene matching, and threat location and coverage. The goal of the planning section is to produce an executable plan of action which can be implemented by the control section. The goal of the control section is also discussed, and an autonomous helicopter mission is described. 86/03/00 86A30488

UTTL: Expert System Pilot Aid
AUTH: A/JURGENSEN, J. R.; B/FELDMANN, R. E. PAA: B/(Systran Corp., Dayton, OH) IN: NAECON 1985; Proceedings of the National Aerospace and Electronics Conference, Dayton, OH, May 20-24, 1985. Volume 2 (A86-28326 12-04). New York, Institute of Electrical and Electronics Engineers, 1985, p. 1583-1590.
ABS: A status report is given for the Expert System Pilot Aid (ESPA) small-business contract with Air Force Avionics Laboratory to study the application of expert system programming techniques to the advanced fighter cockpit. Topics covered include: heuristics vs deterministics, expert system language selection (a modified OPS-5 language is chosen); and mission analysis (emergency scenarios: landing gear failures, engine failure at altitude, system failure during

takeoff). A simulation evaluation of ESPA is to be made using the Avionic System Analysis and Integration Laboratory (AVSAIL) facility at Wright Patterson Air Force Base. 85/00/00 86A28516

UTTL: The Crew Station Information Manager - An avionics expert system
AUTH: A/POHLMANN, L. D.; B/MARKS, P. S.; C/FEHLING, M. R. PAA: A/(Boeing Military Airplane Co., Wichita, KS); C/(Advanced Information and Decision Systems, Mountain View, CA) IN: NAECON 1985; Proceedings of the National Aerospace and Electronics Conference, Dayton, OH, May 20-24, 1985. Volume 2 (A86-28326 12-04). New York, Institute of Electrical and Electronics Engineers, 1985, p. 1576-1582.
ABS: Progress to date is reported on the development of CSIM, the Crew Station Information Manager, which is a prototype avionics expert system being designed and developed under contract to the Air Force Avionics Laboratory. The function of CSIM is to manage the interface between the pilot and the avionics suite of a future tactical aircraft. A review is given of program motivation and objectives, and a set of avionics expert system candidates is identified. Preliminary development and early simulation are discussed, and near-term plans are outlined in connection with a contract extension sponsored by DARPA. 85/00/00 86A28515

UTTL: Path-finder - An hueristic approach to aircraft routing
AUTH: A/LIZZA, C. S.; B/LIZZA, G. PAA: B/(USAF, Wright Aeronautical Laboratories, Wright-Patterson AFB, OH) IN: NAECON 1985; Proceedings of the National Aerospace and Electronics Conference, Dayton, OH, May 20-24, 1985. Volume 2 (A86-28326 12-04). New York, Institute of Electrical and Electronics Engineers, 1985, p. 1436-1443.
ABS: A solution is examined for the problem of developing an effective computer algorithm to route aircraft through hostile enemy defenses. The problem was proposed to assist in studies of the impact of varied, onboard countermeasures upon preplanned aircraft routes. The quality of a route can be defined in terms of the cost of interaction between the aircraft and threats encountered along the path. Current methods of automated routing are either inefficient or produce unsatisfactory results. Beginning with a more general description of the problem, a solution is detailed for developing routes by using the AI technique of heuristic search in an A(asterisk) algorithm. The algorithm uses heuristics based upon estimates of the degree of threat interactivity and distance from a point to the goal. Due to the nature of the heuristic, the A(asterisk) algorithm cannot guarantee development of a least-cost path, but the heuristic can be adjusted to ensure reasonably good results in most cases. The flexibility of the algorithm allows application to the specific aircraft routing problem and to other route planning tasks. 85/00/00 86A28499

UTTL: Knowledge engineering for a flight management expert system
AUTH: A/ANDERSON, B. M.; B/BEAL, J. M.; C/MCNULTY, C.; D/STERN, R. C. PAA: D/(Texas Instruments, Inc., Dallas) IN: NAECON 1985; Proceedings of the National Aerospace and Electronics Conference, Dayton, OH, May 20-24, 1985. Volume 2 (A86-28326 12-04). New York, Institute of Electrical and Electronics Engineers, 1985, p. 1431-1435.
ABS: The development of an F-16 emergency procedures knowledge base for EPES (emergency procedures expert system) is described. A detailed view of the knowledge acquisition problems is presented, followed by a methodology for efficiently building a knowledge base for emergency procedures. A multiple emergency example is considered (loss of canopy and towershaft failure while cruising at 400 knots above 40,000 feet), where the inference engine resolves a conflict in procedures. 85/00/00 86A28498

UTTL: An expert planner for the dynamic flight environment
AUTH: A/CHEN, D. C. PAA: A/(E-Systems, Inc., Garland Div., Dallas, TX) CORP: E-Systems, Inc., Dallas, Tex. IN: NAECON 1985; Proceedings of the National Aerospace and Electronics Conference, Dayton, OH, May 20-24, 1985. Volume 2 (A86-28326 12-04). New York, Institute of Electrical and Electronics Engineers, 1985, p. 1347-1354.
ABS: This paper presents a robust robot planner that functions in the complex and dynamic flight domain. The robot pilot flies an aircraft between two airports and can adjust in flight to changes in the environment such as closed destination airport, thunderstorm in the flight path, and failed engine. The planner adjusts to the world changes by locally patching around the break point instead of complete replanning. The planning architecture is based on the vertical decomposition of domain knowledge, resulting in shallow planning and recovery planning. This robot flight planner can be utilized as the front end of an intelligent flight monitor. The flight planner

dynamically generates the references that are used to determine whether the flight crew should be notified of potential problems. The implementation of this robot planner is also discussed. 85/00/00 86A28490

UTTL: A speech processing system for an air traffic control radar simulator

AUTH: A/LAWSON, R.; B/DAMOULAKIS, J. PAA: A/(Gould, Inc., Simulation Systems Div., Melville, NY); B/(Gould Research Laboratories, Rolling Meadow, IL) IN: NAECON 1985; Proceedings of the National Aerospace and Electronics Conference, Dayton, OH, May 20-24, 1985. Volume 2 (A86-28326 12-04). New York, Institute of Electrical and Electronics Engineers, 1985, p. 987-989.

ABS: A proposed speech control system to replace Track Control Consoles or TCCs (for an air-traffic-control radar simulator) ordinarily operated by pseudo pilots is described. Such Track Control Consoles are manned by personnel who perform the boring task of altering the tracks of simulated aircraft by entering keyboard data in response to the commands from the trainee controller. A speaker-dependent voice recognition system is substituted for the track altering function of each TCC, while a voice synthesis system is substituted for the pilot acknowledgment function. An interface is then created which allows the TCCs to remain in the loop and assume dual aircraft control, should the need arise. A software control system is utilized which permits the rapid movement of voice data from mass storage (on floppy and hard disks) to allow for operation with the required vocabulary. In addition, the relationship between the air traffic control vocabulary and memory requirements is shown, as is the architecture of the proposed system. Finally, the savings in person hours is projected over a typical training scenario for one year of operation. 85/00/00 86A28447

UTTL: An expert system for integrated aircraft/engine controls design

AUTH: A/TAYLOR, J. H. PAA: A/(General Electric Co., Control Technology Branch, Schenectady, NY) IN: NAECON 1985; Proceedings of the National Aerospace and Electronics Conference, Dayton, OH, May 20-24, 1985. Volume 1 (A86-28326 12-04). New York, Institute of Electrical and Electronics Engineers, 1985, p. 661-669.

ABS: In this paper, we discuss the concepts, requirements and architecture for an expert system for integrated aircraft/engine control systems analysis and design. The purpose of this concept is to provide a high-level environment embodying expertise from the many fields that enter into integrated flight and engine controls design (aerodynamics, structures, propulsion, pilot/aircraft interaction, mission performance requirements, control system design and validation), thereby facilitating the design of integrated aircraft control systems and ensuring that the best possible designs are obtained. The expert system concept also has capabilities to handle data-base management, and to take maximum advantage of existing conventional software. A prototype expert system has been implemented that demonstrates many of the capabilities and benefits of such an environment. A record of a hypothetical controls system modeling, analysis, and design session is included in this paper. 85/00/00 86A28405

UTTL: Knowlege-based systems

AUTH: A/DAVIS, R. PAA: A/(MIT, Cambridge, MA) Science (ISSN 0036-8075), vol. 231, Feb. 28, 1986, p. 957-963.

ABS: Consideration is given to the ways in which knowledge-based artificial intelligence (AI) systems have expanded current definitions of the word 'program'. The underlying technology of knowledge-based AI systems are described, with reference made to rule-based systems. The ability of knowledge-based systems to distinguish between what the program is required to know to solve a problem, (the knowledge-base), and the reasoning approach used in the solution of the problem, (the inference engine) is described in detail. The use of knowledge-based AI systems in scientific applications is considered, with emphasis given to applications in aeronautics (the Navex system) chemistry (the Dendral program); medicine; and geology (the Prospector system). 86/02/28 86A27168

UTTL: An expert system for fault tree construction

AUTH: A/GARRIBBA, S.; B/GUAGNINI, E.; C/MUSSIO, P. PAA: B/(Milano, Politecnico, Milan, Italy); C/(Milano, Universita, Milan, Italy) IN: Annual Reliability and Maintainability Symposium, Philadelphia, PA, January 22-24, 1985, Proceedings (A86-22376 08-38). New York, Institute of Electrical and Electronics Engineers, 1985, p. 82-88. *Research supported by the Ministero della Pubblica Istruzione.*

ABS: The architecture of an expert system for the interactive data-driven construction of fault trees is presented. Parts of the system are now under realization and testing. The system intends to offer a flexible and easy-to-operate tool to the analyst in reliability assessment of complex engineered installations. The expert system is organized according to a number of knowledge-based modules that contain metarules, allow to establish rules, and to collect and interpret data. The construction process bases upon a representation of the elementary components given a term of multiple-valued logical (MVL) trees and results into an MVL-tree. This tree can be analyzed directly or when requested it can be reduced to a number of binary fault trees. 85/00/00 86A22383

UTTL: Plessey displays in air traffic control
AUTH: A/FIELD, A. The Controller (ISSN 0010-8073), vol. 24, Dec. 1985, p. 30-32.
ABS: Technological advances which have increased the application of radars to air traffic control are: (1) improved performance of primary radar sensors, (2) the development of a secondary surveillance, (3) digital data processing, and (4) display changes. The design and functions of Watchman, a new generation of autonomous intelligent displays, are examined. The display, which has improved clarity, and the console, which contains a display and data processing system, are integrated. The capabilities the display adds to the radar presentations are listed. The ability of the controller to access all available data as desired is studied. An example of a controller's data display request is provided. 85/12/00 86A21608

UTTL: Development of a knowledge acquisition tool for an expert system flight status monitor
AUTH: A/DISBROW, J. D.; B/DUKE, E. L.; C/REGENIE, V. A. PAA: A/(Systems Control Technology, Inc., Palo Alto, CA); C/(NASA, Flight Research Center, Edwards, CA) CORP: Systems Control Technology, Inc., Palo Alto, Calif.; National Aeronautics and Space Administration. Flight Research Center, Edwards, Calif. AIAA, Aerospace Sciences Meeting, 24th, Reno, NV, Jan. 6-9, 1986. 10 p.
ABS: Two of the main issues in artificial intelligence today are knowledge acquisition and knowledge representation. The Dryden Flight Research Facility of NASA's Ames Research Center is presently involved in the design and implementation of an expert system flight status monitor that will provide expertise and knowledge to aid the flight systems engineer in monitoring today's advanced high performance aircraft. The flight status monitor can be divided into two sections: the expert system itself and the knowledge acquisition tool. This paper discusses the knowledge acquisition tool, the means it uses to extract knowledge from the domain expert, and how that knowledge is represented for computer use. An actual aircraft system has been codified by this tool with great success. Future real-time use of the expert system has been facilitated by using the knowledge acquisition tool to easily generate a logically consistent and complete knowledge base.
RPT#: AIAA PAPER 86-0240 86/01/00 86A19764

UTTL: Artificial intelligence, expert systems, computer vision, and natural language processing
AUTH: A/GEVARTER, W. B. PAA: A/(NASA, Office of Aeronautics and Space Technology, Washington, DC) CORP: National Aeronautics and Space Administration, Washington, D.C. Research sponsored by NASA and NBS. Park Ridge, NJ, Noyes Publications, 1984, 240 p. Previously announced in STAR as N83-31379, N84-10834, and N84-14805.
ABS: An overview of artificial intelligence (AI), its core ingredients, and its applications is presented. The knowledge representation, logic, problem solving approaches, languages, and computers pertaining to AI are examined, and the state of the art in AI is reviewed. The use of AI in expert systems, computer vision, natural language processing, speech recognition and understanding, speech synthesis, problem solving, and planning is examined. Basic AI topics, including automation, search-oriented problem solving, knowledge representation, and computational logic, are discussed. 84/00/00 86A18699

UTTL: A rule-based model of human problem-solving behavior in dynamic environments
AUTH: A/KNAEUPER, A.; B/ROUSE, W. B. PAA: A/(Forschungsinstitut fuer Antrhropotechnik, Wachtberg-Werthhoven, West Germany); B/(Georgia Institute of Technology, Atlanta) IEEE Transactions on Systems, Man, and Cybernetics (ISSN 0018-9472), vol. SMC-15, Nov.-Dec. 1985, p. 708-719.
ABS: Human problem solving is considered with emphasis on situations involving human-machine interaction in detecting, diagnosing, and compensating for failures in engineering systems. An overall model is presented which considers the breath and robustness of human problem-solving behavior in dynamic environments typical of engineering systems. A realization of the general structrue of this model within a particular rule based computer program is discussed. In this program the human behavior in controlling a dynamic process is represented by a set of production rules. The selection of appropriate production rules for a given situation is performed by ordering the rules for

specific tasks and by a control mechanism. The latter provide a means to access only a small part of the knowledge base at a time in order to derive decisions as opposed to searching the whole knowledge base. This program was applied to model human problem solving in a process control task. Results from comparing model and subject behavior are presented and discussed.
85/12/00 86A17772

UTTL: The Autonomous Helicopter System
AUTH: A/GILMORE, J. F. PAA: A/(Georgia Institute of Technology, Atlanta) IN: Applications of artificial intelligence; Proceedings of the Meeting, Arlington, VA, May 3, 4, 1984 (A86-15278 04-63). Bellingham, WA, SPIE - The International Society for Optical Engineering, 1984, p. 146-152.
ABS: This paper describes an autonomous airborne vehicle being developed at the Georgia Tech Engineering Experiment Station. The Autonomous Helicopter System (AHS) is a multimission system consisting of three distinct sections: vision, planning and control. Vision provides the local and global scene analysis which is symbolically represented and passed to planning as the initial route planning constraints. Planning generates a task dependent path for the vehicle to traverse which assures maximum mission system success as well as safety. Control validates the path and either executes the given route or feeds back to previous sections in order to resolve conflicts. 84/00/00 86A15285

UTTL: Meta-level reasoning for scene analysis
AUTH: A/SPIESSBACH, A. J. PAA: A/(Georgia Institute of Technology, Atlanta) IN: Applications of artificial intelligence; Proceedings of the Meeting, Arlington, VA, May 3, 4, 1984 (A86-15278 04-63). Bellingham, WA, SPIE - The International Society for Optical Engineering, 1984, p. 24-30.
ABS: Extending the recent successes demonstrated by artificial intelligence expert system technology to the broader domain of scene analysis necessitates a consequent broadening of the concepts and techniques used in current systems. Simple, single mechanisms must give way to multiple lines of reasoning and multiple levels of description. Meta-level reasoning, which is a recursive application of the basic expert system paradigm, is a promising approach to the problem of coping with the complexity inherent in highly variable, dynamic environments. This paper describes research directed towards incorporating meta-level reasoning into context-based scene analysis systems. A multi-layered expert system architecture is outlined that is aimed at providing high-level strategies and dynamic planning capability to the basic image understanding process. 84/00/00 86A15280

UTTL: Applications of artificial intelligence; *Proceedings of the Meeting, Arlington, VA, May 3, 4, 1984*
AUTH: A/GILMORE, J. F. PAA: A/(Georgia Institute of Technology, Atlanta) Meeting sponsored by SPIE - The International Society for Optical Engineering. Bellingham, WA, SPIE - The International Society for Optical Engineering (SPIE Proceedings. Volume 485), 1984, 243 p. For individual items see A86-15279 to A86-15285.
ABS: Subjects related to expert systems are discussed, taking into account a context dependent automatic target recognition system, computer understanding of air traffic control displays, the role of the image analyst in computer vision, a demonstration of an ocean surveillance information fusion expert system, and the location of multiple faults by diagnostic expert systems. Other topics explored are concerned with knowledge-based systems, autonomous vehicles, and image understanding. Attention is given to aspects of interfacing an intelligent decision-maker to a real-time control system, a reasoning system for computer aided engineering, an 'intelligent' optical design program, an adaptive interpolator algorithm for area-array fine guidance sensors, terrain navigation concepts for autonomous vehicles, the autonomous helicopter system, an autonomous vehicle navigation algorithm, the planning of strategic paths through variable terrain data, the contextual analysis of tactical scenes, and a structural target analysis and recognition system.
RPT#: SPIE-485 84/00/00 86A15278

UTTL: Understanding and automating algorithm design
AUTH: A/KANT, E. PAA: A/(Schlumberger-Doll Research, Ridgefield, CT) IEEE Transactions on Software Engineering (ISSN 0098-5589), vol. SE-11, Nov. 1985, p. 1361-1374.
ABS: Algorithm design is a challenging intellectual activity that provides a rich source of observation and a test domain for a theory of problem-solving behavior. This paper describes a theory of the algorithm design process based on observations of human design and also outlines a framework for automatic design. The adaptation of the theory of human design to a framework for automation in the DESIGNER system leads to a better understanding of

human design, and the implementation process helps validate the framework. Issues discussed in this paper include the problem spaces used for design, the loci of knowledge and problem-solving power, and the relationship to other methods of algorithm design and to automatic programming as a whole.   85/11/00
86A14848

UTTL: Enhanced maintenance and explanation of expert systems through explicit models of their development
AUTH: A/NECHES, R.; B/SWARTOUT, W. R.; C/MOORE, J. D. PAA: C/(Southern California, University, Marina del Rey)   IEEE Transactions on Software Engineering (ISSN 0098-5589), vol. SE-11, Nov. 1985, p. 1337-1351.
ABS: Principled development techniques could greatly enhance the understandability of expert systems for both users and system developers. Current systems have limited explanatory capabilities and present maintenance problems because of a failure to explicitly represent the knowledge and reasoning that went into their design. This paper describes a paradigm for constructing expert systems which attempts to identify that tacit knowledge, provide means for capturing it in the knowledge bases of expert systems, and apply it towards more perspicuous machine-generated explanations and more consistent and maintainable system organization.   85/11/00
86A14847

UTTL: Uncertainty and control; Proceedings of the International Seminar, Bonn, West Germany, May 1985
AUTH: A/ACKERMANN, J.   PAA: A/(DFVLR, Institut fuer Dynamik der Flugsysteme, Wessling, West Germany)   Seminar sponsored by DFVLR. Berlin and New York, Springer-Verlag (Lecture Notes in Control and Information Sciences. Volume 70), 1985, 240 p. For individual items see A86-14827 to A86-14833.
ABS: The state of the art in the theory of the design of general dynamic systems is surveyed, stressing uncertainty and control. The relationship of theory with applied engineering problems and solutions is addressed. The topics discussed include: DFVLR activities concerning uncertainty and control, system identification, uncertainty models and the design of robust control systems, multimodel approaches to robust control system design, adaptive control as a way to deal with uncertainty, optimality in adaptive control systems, and intelligent control-operating systems in uncertain environments   85/00/00
86A14826

UTTL: The blackboard model - A framework for integrating multiple cooperating expert systems
AUTH: A/ERICKSON, W. K.   PAA: A/(NASA, Ames Research Center, Moffett Field, CA)   CORP: National Aeronautics and Space Administration. Ames Research Center, Moffett Field, Calif.   IN: Computers in Aerospace Conference, 5th, Long Beach, CA, October 21-23, 1985, Technical Papers (A86-11401 02-59). New York, AIAA, 1985, p. 33-40.
ABS: The use of an artificial intelligence (AI) architecture known as the blackboard model is examined as a framework for designing and building distributed systems requiring the integration of multiple cooperating expert systems (MCXS). Aerospace vehicles provide many examples of potential systems, ranging from commercial and military aircraft to spacecraft such as satellites, the Space Shuttle, and the Space Station. One such system, free-flying, spaceborne telerobots to be used in construction, servicing, inspection, and repair tasks around NASA's Space Station, is examined. The major difficulties found in designing and integrating the individual expert system components necessary to implement such a robot are outlined. The blackboard model, a general expert system architecture which seems to address many of the problems found in designing and building such a system, is discussed. A progress report on a prototype system under development called DBB (Distributed BlackBoard model) is given. The prototype will act as a testbed for investigating the feasibility, utility, and efficiency of MCXS-based designs developed under the blackboard model.
RPT#: AIAA PAPER 85-5045   85/00/00   86A11407

UTTL: R1-Soar - An experiment in knowledge-intensive programming in a problem-solving architecture
AUTH: A/ROSENBLOOM, P. S.; B/LAIRD, J. E.; C/MCDERMOTT, J.; D/NEWELL, A.; E/ORCIUCH, E.   PAA: A/(Stanford University, CA); B/(Xerox Research Center, Palo Alto, CA); D/(Carnegie-Mellon University, Pittsburgh, PA); E/(Digital Equipment Corp., Maynard, MA)   IEEE Transactions on Pattern Analysis and Machine Intelligence (ISSN 0162-8828), vol. PAMI-7, Sept. 1985, p. 561-569. Research supported by the Digital Equipment Corp.
ABS: This paper presents an experiment in knowledge-intensive programming within a general problem-solving production-system architecture called Soar. In Soar, knowledge is encoded within a set of problem spaces, which yields a system capable of reasoning from first principles. Expertise consists of additional rules that guide complex problem-space searches and substitute for expensive problem-space

operators. The resulting system uses both knowledge and search when relevant. Expertise knowledge is acquired either by having it programmed, or by a chunking mechanism that automatically learns new rules reflecting the results implicit in the knowledge of the problem spaces. The approach is demonstrated on the computer-system configuration task, the task performed by the expert system R1. 85/09/00
85A48596

UTTL: Enhancing knowledge representation in engineering databases
AUTH A/HARTZBAND, D. J.; B/MARYANSKI, F. J. PAA: A/(Digital Equipment Corp., Maynard, MA); B/(Connecticut, University, Storrs) Computer (ISSN 0018-9162), vol. 18, Sept. 1985, p. 39-46. Research supported by Digital Equipment Corp.
ABS: Recent advances in database management (DBM) technologies pertinent to the support of engineering requirements are reviewed. Engineering databases involve a variety of data types, the ability to express complex relationships between model elements, and knowledge (as well as data) representation. Data models are constructed from object types, operators and integrity rules, yielding formal tools for representing and manipulating information. Current model development efforts are concentrating on enhancing the capabilities for defining logical relationships among data elements, for data abstraction, for describing properties, operations and constraints on each type of object, and to expand to different types of data objects such as images. Knowledge base management relies on extracting meaning from data. Its implementation depends on the further development of rule- or frame-based problem solving within the DBM system. Techniques being pursued to obtain an ideal data mode which balances between naturalness and ease of expression and an adequate range of expression are explored. 85/09/00
85A48022

UTTL: 1984 American Control Conference, San Diego, CA, June 6-8, 1984, Proceedings. Volumes 1, 2 & 3 Conference sponsored by the American Automatic Control Council. New York, IEEE, 1984, Vol. 1, 623 p.; vol. 2, 628 p.; vol. 3, 772 p. For individual items see A85-47677 to A85-47803.
ABS: The topics considered are related to the modeling of human cognitive decision processes, sensor-based robot control systems, adaptive control and applications, modelling and simulation of thermofluid processes and systems, advanced concepts for computer-aided control system design, model reduction and large scale systems, fuel-optimal aircraft guidance and control, and digital signal processing. Other subjects explored are concerned with the dynamical systems approach to problems in nonlinear systems and control, monitoring and fault detection in power systems, robot path planning and control, the real time control of processes, pole placement design, large scale systems and model reduction, and aircraft control. Attention is also given to servomechanisms and machine tool control, stochastic systems, process model-based control and analysis, applications of multivalued logic, microprocessor implementation of real time control systems using high order languages, multitarget tracking, digital systems, filtering and estimation, optimal control, and fault tolerant aerospace systems. 84/00/00 85A47676

UTTL: Design of an expert-system flight status monitor
AUTH A/REGENIE, V. A.; B/DUKE, E. L. PAA: B/(NASA, Flight Research Center, Edwards, CA) CORP: National Aeronautics and Space Administration. Flight Research Center, Edwards, Calif. AIAA, Guidance, Navigation and Control Conference, Snowmass, CO, Aug. 19-21, 1985. 9 p.
ABS The present technology used to monitor systems in flight tests is not advanced enough for the modern avionics in high performance aircraft. Research is being conducted at NASA's Dryden Flight Research Facility to design an expert system to monitor test flights. The expert system is to automatically detect any problems in the flight control system (FCS), interpret the problem from the information contained in its knowledge base, inform the systems engineer, and recommend solutions. The data is to be downlinked from the aircraft to the control room. The expert system will lessen the responsibilities of the engineers by providing them with fast, expert advice. Time is the most critical factor in flight testing and the expert system will be able to quickly recognize discrepancies and provide corrections. A demonstration of the expert system, not operating in real time, has already been tested.
RPT#: AIAA PAPER 85-1908 85/08/00 85A45975

UTTL: Combining quantitative and qualitative reasoning in aircraft failure diagnosis
AUTH A/STENGEL, R. F.; B/HANDELMAN, D. A. PAA: A/(Princeton University, NJ) IN: Guidance, Navigation and Control Conference, Snowmass, CO, August 19-21, 1985, Technical Papers (A85-45876 22-08). New York, AIAA, 1985, p. 366-375

ABS The *problem of in-flight failure origin diagnosis* is addressed by combining aspects of analytical redundancy and artificial intelligence theory. The objective is to use the mathematical model designed to simulate aircraft behavior as a *supplement to the* knowledge used for diagnosis. A method is developed whereby qualitative causal information about a dynamic system is drawn from its model. Based on sensitivities of the equations of motion to worst-case failure modes, a measure of the relative capacity of system elements to affect one another is derived. A diagnosis procedure combining problem reduction and backward-chaining ordered search uses this knowledge to reduce a list of elements capable of failure to a relatively small list of elements *suspected of* failure. Examples illustrate use of the knowledge base and the problem-solving mechanism that has been developed. Two parameters are found to be crucial to the fault diagnosis: the elapsed time between first detection of a failure and initiation of the diagnosis procedure, and the minimum amount of influence that an element must have on a well-behaved indicator in order to deem the element unfailed.

RPT#: AIAA PAPER 85-1905 85/00/00 85A45917

UTTL: Towards an expert system architecture *for flight domain applications*

AUTH: A/CROSS, S. E. PAA: A/(USAF, Institute of Technology, Wright-Patterson AFB, OH) IN: NAECON 1984; Proceedings of the National Aerospace and Electronics Conference, Dayton, OH, May 21-25, 1984. Volume 2 (A85-44976 21-01). New York, IEEE, 1984, p. 784-788.

ABS: Several limitations in the application of basic rule-based systems to the flight domain are discussed. Some potential applications of expert systems in combat aircraft automation are summarized, and the paradigm of a rule-based system, which is essentially a production system with a particular knowledge organization and strategy for knowledge utilization, is examined. Knowledge representation in expert systems is considered along with the role of metaplaning, which deals with conflicting goals, in the flight domain. Qualitative reasoning, which draws conclusions from incomplete observations and knowledge, is discussed, and the necessity for pilot aid access to models of the systems in which they are experts is emphasized. 84/00/00 85A45084

UTTL: NAECON 1984; Proceedings of the National Aerospace and Electronics Conference, Dayton, OH, May 21-25, 1984. Volumes 1 & 2. Conference sponsored by IEEE. New York, IEEE, 1984. Vol. 1, 736 p.; vol. 2, 768 p. For individual items see A85-44977 to A85-45161.

ABS: Developments related to VLSI are discussed along with topics concerned with signal processing, cartographic data uses, *data transmission*, avionics system topics, multiapplication signal processing architectures, airborne image processing, target recognition/acquisition, airborne radar and fire control, navigation, air data, weapon guidance and control, Kalman filtering, power generation and control, and flying qualities. Attention is given to integrated control, flight management, multivariable control, self repairing flight control, all-electric aircraft, digital flight control architecture and reliability, advanced software tools, software acquisition and test issues, software management and quality assurance, expert systems, trends in artificial intelligence, and engineering management. Other areas considered include system performance and workload assessment, human/machine system analysis, advanced avionics display content, reliability, life cycle cost, and flight training and simulation. 84/00/00 85A44976

UTTL: Generative engineering databases - Toward expert systems

AUTH: A/RASDORF, W. J.; B/SALLEY, G. C. PAA: A/(North Carolina State University, Raleigh); B/(NASA, Langley Research Center, Hampton, VA) CORP: North Carolina State Univ., Raleigh.; National Aeronautics and Space *Administration*. Langley Research Center, Hampton, Va. (George Washington University and NASA, Symposium on Advances and Trends in Structures and Dynamics, *Washington*, DC, Oct. 22-25, 1984) Computers and *Structures* (ISSN 0045-7949), vol. 20, no. 1-3, 1985, p. 11-15.

ABS: Engineering data management, incorporating concepts of optimization with data representation, is receiving increasing attention as *the amount and complexity* of information necessary for performing engineering operations increases and the need to coordinate its representation and use increases. Research in this area promises advantages for a wide variety of engineering applications, particularly those which seek to use data in innovative ways in the engineering process. *This paper presents* a framework for a comprehensive, relational database management system that combines a knowledge base of design constraints with a database of engineering data items in order to achieve a 'generative database' - one which automatically generates new engineering design data according to the design constraints stored in the knowledge base. The representation requires a database that is able to store all of the data normally associated with engineering design and to accurately represent the interactions between constraints and the stored data while guaranteeing its integrity. The representation also requires a knowledge base that is able to store all the constraints imposed upon the engineering design process. 85/00/00 85A41103

UTTL: Invincible aircraft may be a step closer to reality
AUTH: A/HOLT, D. J. Aerospace Engineering (ISSN 0002-1458), vol. 5, Jan. 1985, p. 8-11.
ABS: A discussion is presented concerning the development status and prospective performance enhancements available through 'self-repairing' electronic flight control systems for military aircraft. Such systems would be able to significantly reduce battlefield vulnerability by automatically compensating for damage incurred in one aircraft structure or component by altering the performance of others. Attention is given to the configurational, software logic, and technology validation requirements of such systems. 85/01/00 85A36723

UTTL: Model-based reasoning in expert systems - An application to enroute air traffic control
AUTH: A/CROSS, S. E. PAA: A/(USAF, Institute of Technology, Wright-Patterson AFB, OH) IN: Digital Avionics Systems Conference, 6th, Baltimore, MD, December 3-6, 1984, Proceedings (A85-17801 06-01). New York, American Institute of Aeronautics and Astronautics, 1984, p. 95-101. USAF-sponsored research, U.S. Department of Transportation.
ABS: The explanation capabilities (EC) of expert systems, the extent of computer understanding, and the artificial intelligence ability to reason about disparate knowledge are discussed in the context of air traffic control (ATC). EC is essential for humans to understand and interact with the results of computer reasoning. Questions of 'how' and 'why' certain actions are recommended can be satisfied by a display of the appropriate part of the computational process used to arrive at a conclusion, abstracted and expressed in a form amenable to the context of the question and intelligible to humans. The knowledge base may be solutions to the aircraft equations of motion. It may be necessary for representations to be multi-leveled to reply successively until satisfying the questioner's level of sophistication in understanding, e.g., physics. For ATC problems such as collision avoidance, the system must take into account operational aspects like other flight routes and flight economy. Several examples are provided of means by which an expert system could search for an answer and be able to explain it.
RPT#: AIAA PAPER 84-2619 84/00/00 85A17817

UTTL: Artificial intelligence implications for advanced pilot/vehicle interface design
AUTH: A/MAXWELL, K. J.; B/DAVIS, J. A. PAA: B/(General Dynamics Corp., Fort Worth, TX) IN: Digital Avionics Systems Conference, 6th, Baltimore, MD, December 3-6, 1984, Proceedings (A85-17801 06-01). New York, American Institute of Aeronautics and Astronautics, 1984, p. 90-94.
ABS: The impact on pilot/vehicle interface (PVI) design for fighter aircraft from the introduction of artificial intelligence (AI) technology is discussed. Three prototypical models (pilot manager/AI associate, pilot/AI colleague, Autonomous assistant) of the operational relationship between the pilot and AI systems are defined. These models provide a structure in which PVI issues are discussed. Issues involving the resolution of possible disagreements between the pilot and the AI system, intelligent presentation of information including an intelligent interrupt capability, and natural language interaction are discussed. It is concluded that the introduction of AI into the aircraft will have a major impact on PVI design.
RPT#: AIAA PAPER 84-2617 84/00/00 85A17816

UTTL: Flight control technology for improved aircraft performance and survivability
AUTH: A/GAYLOR, R. PAA: A/(Sperry Corp., Defense Systems Div., Albuquerque, NM) AIAA, AHS, and ASEE, Aircraft Design Systems and Operations Meeting, San Diego, CA, Oct. 31-Nov. 2, 1984. 7 p.
ABS: The state-of-the-art in flight control is examined, including the historical development and future prospects. A general fly-by-wire system is discussed with regard to its reliability and backup level requirement. The advantages of fiber optics are discussed including its immunity to lightning and nuclear electromagnetic pulse, and its ability to provide high-speed communications between subsystems and channels of the redundant flight control system. Furthermore, flight control system actuators are discussed, including conventional low-torque value; Quadruplex Electronics Signal/Dual Hydraulic Force

Motor Control Value; Quadruplex Actuator with Fiber Optics Interfaces; and Dual Electric Actuator which contains no hydraulic or pneumatic system. Finally, the prospects of applying artificial intelligence to flight control systems are considered. Block diagrams and schematic drawings are included.
RPT#: AIAA PAPER 84-2489 84/10/00 85A16107

UTTL: Artificial intelligence applied to the inertial navigation system performance and maintenance improvement
AUTH: A/BEDOYA, C. A.; B/KELLER, K. J. PAA: B/(McDonnell Aircraft Co., St. Louis, MO) IN: Institute of Navigation, Annual Meeting, 40th, Cambridge, MA, June 25-28, 1984 (A85-14826 04-04). Washington, DC, Institute of Navigation, 1984, p. 55-61.
ABS: Test results and performance and design features of an inertial navigation system-fault analysis and management system (INS-FAAMS) for implementing artificial intelligence (AI) in fighter aircraft avionics are outlined. Development objectives of INS-FAAMS were to enhance the availability and accuracy of INS and demonstrate AI expert system capabilities. INS failures were studied and found to be intermittent, induced by an incorrect procedure, or due to nonduplicable conditions. The expert system AI data base was defined through field maintenance data, identification of key failure paths, and simulation testing with regards different mission profiles. A blackboard architecture was selected and comprised three divisions, permanent knowledge and current hypothesis, knowledge source demons searching for an antecedent to become true, and a priority-based scheduler. Tests have revealed the INS-FAAMS effectiveness at identifying problems for maintenance or isolating them when possible. 84/00/00 85A14830

UTTL: Institute of Navigation, Annual Meeting, 40th, Cambridge, MA, June 25-28, 1984 Meeting sponsored by ION, General Motors Corp., Northrop Corp., et al. Washington, DC, Institute of Navigation, 1984, 177 p. For individual items see A85-14827 to A85-14841.
ABS: Existing, planned, and prototype systems for aiding land, marine, and aircraft navigation, positioning, and landing approaches are described. Attention is given to navigation aid data processing in the 737-300 FMC, the accuracy of OMEGA navigation system position fix, and features of the planned Geostar commercial navigation and communications system. The implementation of artificial intelligence expert systems in fighter avionics is discussed, along with inertial sensor systems, suboptimal filtering of GPS signals, and VLF radio compass for Arctic navigation. Clock coasting and error analysis, a pseudo-satellite beacon system, a receiver processor, and a data base for the GPS navigation system are considered. Finally, a low-cost GPS receiver/processor is detailed and an integrated surveillance/navigation system is outlined. 84/00/00 85A14826

UTTL: BIT false alarms - An important factor in operational readiness
AUTH: A/MALCOLM, J. G. PAA: A/(Hughes Aircraft Co., Canoga Park, CA) In: Annual Reliability and Maintainability Symposium, Los Angeles, CA, January 26-28, 1982, Proceedings. (A82-42176 21-38) New York, Institute of Electrical and Electronics Engineers, 1982, p. 206-212.
ABS: The premise of this paper is that current avionic systems are inherently reliable and potentially maintainable at high rates of operational readiness (OR). OR rates are frequently less than anticipated because of excessive maintenance. That much maintenance is unnecessary is evidenced in part by the problem of high removal rate of fault-free units reported by military test and evaluation organizations. This problem is very complex, including such things as level of training of maintenance personnel. The problem is partly that built in test (BIT) yields indications which do not represent actual failures, i.e., are false alarms. This paper describes some root causes of false alarms and identifies solution approaches, including improved specifications, improved analysis techniques, and expanded use of new technology. Resolving the false alarm problem can result in a major improvement in operational readiness. 82/00/00 82A42193

UTTL: PROLOG: An expert system building tool
AUTH: A/VANDERGAAG, L. C. CORP: Center for Mathematics and Computer Science, Amsterdam (Netherlands). CSS: (Dept. of Computer Science/Software Technology.)
ABS: The suitability of PROLOG as an expert system building tool is demonstrated. A small expert system shell is discussed, and compared to the DELFI-2 system. Advantages of PROLOG include the separation of knowledge and inference as a principle in the development of expert system shells, achieved naturally as a consequence of the principles of logic programming. Production rules are represented as Horn clauses in a straightforward manner. Part of the inference engine is already provided for in the PROLOG interpreter. The use of a PROLOG expert system in a

real-life environment, however, has a considerable drawback: its inefficiency compared with, for instance, Pascal. Nonetheless, a PROLOG rule based expert system can be developed in a short time, making PROLOG a suitable language for the rapid development of propotype systems.

RPT#: CWI-CS-R8616 B8671572 ETN-86-98674 86/04/00 87N14893

UTTL: A general object-centered database language (GODEL): A preliminary definition

AUTH: A/KERSTEN, M. L.; B/SCHIPPERS, F. H. CORP: Center for Mathematics and Computer Science, Amsterdam (Netherlands). CSS: (Dept. of Computer Sci./Algorithms and Architecture.)

ABS: The programming language GODEL, for the construction of knowledge based applications is described. It uses object-centered, rule-oriented, and procedural programming paradigms. These paradigms are used in an unconventinal way, thereby simplifying the maintenance of complex relationships among (static) objects and modeling dynamic behavior through actor-like objects, called guardians. An informal semantic definition is given. The language definition assumes a teletype-like user interface, which simplifies the language specification and its implementation. A functional prototype was implemented in C-PROLOG under UNIX BSD4.2.

RPT#: CWI-CS-R8615 B8671571 ETN-86-98673 86/04/00 87N14892

UTTL: Knowledge representation and inference in rule-based systems

AUTH: A/LUCAS, P. J. F. CORP: Center for Mathematics and Computer Science, Amsterdam (Netherlands). CSS: (Dept. of Computer Science/Software Technology.)

ABS: Approaches to representing and applying human knowledge in expert systems, in particular in rule-based systems, are reviewed. Equivalent methods of representation are introduced. Low-level operations and inference procedures applied in extracting useful knowledge from a knowledge base are emphasized. The design and the implementation of the DELFI-2 system, particularly influenced by concepts from logic programming, are mentioned.

RPT#: CWI-CS-R8613 B8671569 ETN-86-98671 86/04/00 87N14891

UTTL: Rapid prototyping facility for flight research in artificial-intelligence-based flight systems concepts

AUTH: A/DUKE, E. L.; B/REGENIE, V. A.; C/DEETS, D. A. CORP: National Aeronautics and Space Administration. Ames Research Center, Moffett Field, Calif.

ABS: The Dryden Flight Research Facility of the NASA Ames Research Facility of the NASA Ames Research Center is developing a rapid prototyping facility for flight research in flight systems concepts that are based on artificial intelligence (AI). The facility will include real-time high-fidelity aircraft simulators, conventional and symbolic processors, and a high-performance research aircraft specially modified to accept commands from the ground-based AI computers. This facility is being developed as part of the NASA-DARPA automated wingman program. This document discusses the need for flight research and for a national flight research facility for the rapid prototyping of AI-based avionics systems and the NASA response to those needs.

RPT#: NASA-TM-88268 H-1367 NAS 1.15:88268 86/10/00 87N12273

UTTL: *On the utilization of engineering* knowledge in design optimization

AUTH: A/PAPALAMBROS, P. CORP: Michigan Univ., Ann Arbor. CSS: (Dept. of Mechanical Engineering.)

ABS: Some current research work conducted at the University of Michigan is described to illustrate efforts for incorporating knowledge in optimization in a nontraditional way. The incorporation of available knowledge in a logic structure is examined in two circumstances. The first examines the possibility of introducing global design information in a local active set strategy implemented during the iterations of projection-type algorithms for nonlinearly constrained problems. The technique used algorithms for nonlinearly constrained problems. The technique used combines global and local monotinicity analysis of the objective and constraint functions. The second examines a knowledge-based program which aids the user to create condigurations that are most desirable from the manufacturing assembly viewpoint. The data bank used is the classification scheme suggested by Boothroyd. The important aspect of this program is that it is an aid for synthesis intended for use in the design concept phase in a way similar to the so-called idea-triggers in creativity-enhancement techniques like brain-storming. The idea generation, however, is not random but it is driven by the goal of achieving the best acceptable configuration.
84/00/00 87N11777

UTTL: The representation of knowledge for situations and events in a dynamic world
AUTH: A/SCHMIEDEL, A.; B/VONLUCK, K. CORP: Technische Univ., Berlin (West Germany).
ABS: The representation of conceptual knowledge in order to construct an artificial epistemic subject using artificial intelligence is discussed. The emphasis is on a theoretical foundation for artificial systems which allow information exchange with a human user. The requirements for such a system are given. Formalisms for the representation of partial knowledge using abstract concepts and the dynamic construction of knowledge bases are presented. 85/12/00 86N32128

UTTL: A.I.-based real-time support for high performance aircraft operations
AUTH: A/VIDAL, J. J. CORP: California Univ., Los Angeles. CSS: (Dept. of Computer Science.)
ABS: Artificial intelligence (AI) based software and hardware concepts are applied to the handling system malfunctions during flight tests. A representation of malfunction procedure logic using Boolean normal forms are presented. The representation facilitates the automation of malfunction procedures and provides easy testing for the embedded rules. It also forms a potential basis for a parallel implementation in logic hardware. The extraction of logic control rules, from dynamic simulation and their adaptive revision after partial failure are examined. It uses a simplified 2-dimensional aircraft model with a controller that adaptively extracts control rules for directional thrust that satisfies a navigational goal without exceeding pre-established position and velocity limits. Failure recovery (rule adjusting) is examined after partial actuator failure. While this experiment was performed with primitive aircraft and mission models, it illustrates an important paradigm and provided complexity extrapolations for the proposed extraction of expertise from simulation, as discussed. The use of relaxation and inexact reasoning in expert systems was also investigated.
RPT#: NASA-CR-176906 NAS 1.26:176906 85/00/00 86N30718

UTTL: Development experience with a simple expert system demonstrator for pilot emergency procedures
AUTH: A/VANNORMAN, M.; B/MACKALL, D. A. CORP: National Aeronautics and Space Administration. Ames Research Center, Moffett Field, Calif.
ABS: Expert system techniques, a major application area of artificial intelligence (AI), are examined in the development of pilot associate to handle aircraft emergency procedures. The term pilot associate is used to describe research involving expert systems that can assist the pilot in the cockpit. The development of expert systems for the electrical system and flight control system emergency procedures are discussed. A simple, high-level expert system provides the means to choose which knowledge domain is needed. The expert systems were developed on a low-cost, FORTH-based package, using a personal computer.
RPT#: NASA-TM-85919 H-1272 NAS 1.15:85919 86/02/00 86N23603

UTTL: Implementation of artificial intelligence rules in a data base management system
AUTH: A/FEYOCK, S. CORP: VAIR, Inc., Williamsburg, Va.
ABS: The intelligent front end prototype was transformed into a RIM-integrated system. A RIM-based expert system was written which demonstrated the developed capability. The use of rules to produce extensibility of the intelligent front end, including the concept of demons and rule manipulation rules were investigated. Innovative approaches such as syntax programming were to be considered.
RPT#: NASA-CR-178048 NAS 1.26:178048 86/02/00 86N21220

UTTL: Space station data system analysis/architecture study. Task 5: Program plan CORP: McDonnell-Douglas Astronautics Co., Huntington Beach, Calif.
ABS: Cost estimates for both the on-board and ground segments of the Space Station Data System (SSDS) are presented along with summary program schedules. Advanced technology development recommendations are provided in the areas of distributed data base management, end-to-end protocols, command/resource management, and flight qualified artificial intelligence machines.
RPT#: NASA-CR-177846 NAS 1.26:177846 MDC-H1343A 85/12/00 86N20481

UTTL: *Databases and automated reasoning*
AUTH: A/LUSK, E. L.; B/OVERBEEK, R. A. CORP: Argonne National Lab., Ill. Presented at the International Topical Meeting on Computer Applications for Nuclear Power Plant Operation and Control, Pasco, Wash., 8 Sep. 1985
ABS: The design of application systems incorporating automated reasoning technology requires an understanding of database design, familiarity with various types of automated reasoning methods, and an integrated view of the database and reasoning system.

We describe here the relationship, based on the common language of first-order logic, between entity-relationship database design techniques and automated reasoning systems. We include a brief overview of logic programming, automated theorem proving, and production-rule systems as separate but related branches of automated reasoning.
RPT#: DE85-018348 CONF-850903-17 85/00/00 86N18247

UTTL: Microelectronics technology
AUTH: A/SIDOROVYY, M. CORP: Joint Publications Research Service, Arlington, Va.
ABS: To develop at a leading pace the production of high-speed control and computational systems, peripheral equipment and software for these, electronic devices for control and telemechanics applications, a nation must have a strong microelectronics industry. The world of microelectronics created by man is one of fantastic possibilities and its horizons broaden daily. It is undergoing a process of interlocking growth and of introduction into our daily lives as well as into many fields of human endeavor. Microelectronics is a catalyst for technical progress. It provides artificial intelligence for robots and systems of robots, for automated systems used in control, teaching, design and computing applications, spacecraft and production facilities. A country's economic might and defensive capability are defined by the scale at which it produces pure and ultrapure substances and materials, including semiconductor and metallic crystals. 85/08/27 86N15930

UTTL: Pilot's associate definition study
AUTH: A/EISENHARDT, R. G.; B/EISENHARDT, G. H.; C/DOUTHAT, D. Z. CORP: Perceptronics, Inc., Ann Arbor, Mich.
ABS: Air combat operational and technological problems in the 1995 technology time frame are analyzed for application of artificial intelligence technology under the DARPA's Strategic Computing Program. Operational missions are examined to identify those functions most critical for mission success. Applicable technology is reviewed including sources other than the SCP. A case is made for application of information processing architectural concepts new to airborne weapon systems, but matured in other disciplines over several decades, to accommodate AI technology. Such architecture are developed and reviewed. Onboard information processing hierarchies are developed to support mission requirements for both air-to-air and air-to-ground missions. These are then grouped into five processors, including a pilot/vehicle interface, to transition from the output of sensor signal processing to highly symbolic information for presentation to the pilot/crew. These processors are a Situation Awareness Manager to develop and maintain a current world model external to the airplane, a System Status Manager to perform the same function for the world inside the plane, a Tactical Manager to assess tactical implications of the world and make recommendations to the pilot for action, and a Mission Manager to evaluate mission plans vis-a-vis current situation and assist the pilot in revising plans as necessary. Artificial intelligence technology, both hardware and software, is reviewed and areas and degree of application efforts, measures of effectiveness (MOE) and performance (MOP) are suggested and defined.
RPT#: AD-A157106 FR2213U/4577 85/05/00 86N13026

UTTL: Description of an experimental expert system flight status monitor
AUTH: A/DUKE, E. L.; B/REGENIE, V. A. CORP: National Aeronautics and Space Administration. Ames Research Center, Moffett Field, Calif. Presented at the 5th AIAA Computers in Aerospace Conf., Long Beach, Calif., 21-23 Oct. 1985
ABS: This paper describes an experimental version of an expert system flight status monitor being developed at the Dryden Flight Research Facility of the NASA Ames Research Center. This experimental expert system flight status monitor (ESSFSM) is supported by a specialized knowledge acquisition tool that provides the user with a powerful and easy-to-use documentation and rule construction tool. The EESFSM is designed to be a testbed for concepts in rules, inference mechanisms, and knowledge structures to be used in a real-time expert system flight status monitor that will monitor the health and status of the flight control system of state-of-the-art, high-performance, research aircraft.
RPT#: NASA-TM-86791 H-1317 NAS 1.15:86791 AIAA-85-6042-CP 85/10/00 86N11195

UTTL: A database design for the Brazilian Air Force Flying Unit Operational Control System
AUTH: A/DACUNHA, A. M. CORP: Air Force Inst. of Tech., Wright-Patterson AFB, Ohio. CSS: (School of Engineering.)
ABS: This thesis addresses a relational database design for a Brazilian Air Force Flying Unit Operational Control System. After defining the problem and specifying requirements, an overall system analysis was performed using Decision Support System Theory. A top-down

planning for decision support systems and databases, and a functional analysis were performed to identify potential environmental database applications. Using a canonical approach, a file management systme was mapped to a database conceptual model and afterwards to a database logical model. A prototype Dialog Generator Management Software was implemented through menu driven programs, using the dBASE II DBMS version 2.1 running on a Z-80 microcomputer. The database partial implementation was performed using the INGRES DBMS version 7.10 running on a VAX 11/780, from which advanced queries were retrieved. Finally, an investigation of optimum query retrievals from databases is performed using Artificial Intelligence methods and techniques

RPT# AD-A151848 AFIT/GCS/ENG/84D-7 84/12/14 85N26452

UTTL: Automatic control design procedures for restructurable aircraft control

AUTH: A/LOOZE, D. P.; B/KROLEWSKI, S.; C/WEISS, J.; D/BARRETT, N.; E/ETERNO, J. CORP: Alphatech, Inc., Burlington, Mass.

ABS: A simple, reliable automatic redesign procedure for restructurable control is discussed. This procedure is based on Linear Quadratic (LQ) design methodologies. It employs a robust control system design for the unfailed aircraft to minimize the effects of failed surfaces and to extend the time available for restructuring the Flight Control System. The procedure uses the LQ design parameters for the unfailed system as a basis for choosing the design parameters of the failed system. This philosophy alloys the engineering trade-offs that were present in the nominal design to the inherited by the restructurable design. In particular, it alloys bandwidth limitations and performance trade-offs to be incorporated in the redesigned system. The procedure also has several other desirable features. It effectively redistributes authority among the available control effectors to maximize the system performance subject to actuator limitations and constraints. It provides a graceful performance degradation as the amount of control authority lessens. When given the parameters of the unfailed aircraft, the automatic redesign procedure reproduces the nominal control system design

RPT# NASA-CR-172489 NAS 1.26:172489 TR-212-1 85/01/00 85N21173

UTTL: Intelligent interfaces for tactical airborne platforms

AUTH: A/MADNI, A. CORP: National Aeronautics and Space Administration. Ames Research Center, Moffett Field, Calif.

ABS: Enhanced capabilities of tactical airborne platforms have resulted in increased r mber of aircrew tasks, greater task complexity, and increased time-stress in task performance. Embedded intelligence in the aircrew-vehicle interface (AVI) can help alleviate aircrew workload and enhance aircrew performance by: (1) optimizing the exchange of information between the aircrew and the onboard automation; and (2) adaptively allocating functions between aircrew and automation in response to situational demands. Intelligent interface issues are addressed in this report such as: (1) how to ensure that the aircrew can cope with the information influx; (2) how to present/portray both situational and internal status information; (3) how to allocate functions between the aircrew and the onboard automation; and (4) how to explain reasoning processes employed by onboard intelligence to the aircrew. 84/12/00 85N14820

UTTL: Artificial intelligence theory and reconfigurable control systems

AUTH: A/STENGEL, R. F. CORP: Princeton Univ., N. J. CSS (Dept. of Mechanical and Aerospace Engineering.)

ABS: A program for the analytic and experimental investigation of reconfigurable control systems is described. Its principal objectives are to extend the theory of artificial intelligence and to develop practical methods of applying artificial intelligence heuristics, tatistical hypothesis testing, and modern control theory to the reconfiguration of control systems following sensor failures, actuator failures, power supply or transmission failures, or unforeseen changes in dynamic characteristics. Objectives include the definition of typical failure modes and effects; formulation and investigation of algorithms for detection, identification, estimation, and control; numerical simulation of failure and reconfiguration, and experimentation using a microprocessor-based reconfigurable control system.

RPT# AD-A143766 MAE-1664 ARO-20155.1-MA 84/06/30 84N34203

REPORT DOCUMENTATION PAGE

1. Recipient's Reference	2. Originator's Reference	3. Further Reference	4. Security Classification of Document
	AGARD-LS-155	ISBN 92-835-1588-7	UNCLASSIFIED

5. Originator	Advisory Group for Aerospace Research and Development North Atlantic Treaty Organization 7 rue Ancelle, 92200 Neuilly sur Seine, France
6. Title	KNOWLEDGE BASED CONCEPTS AND ARTIFICIAL INTELLIGENCE APPLICATIONS TO GUIDANCE AND CONTROL
7. Presented on	10—11 September 1987 in Ottawa, Canada, 14-15 September 1987 in Monterey, USA, 12—13 October 1987 in Delft, The Netherlands and 15—16 October 1987 in Lisbon, Portugal.

8. Author(s)/Editor(s)	9. Date
Various	August 1987

10. Author's/Editor's Address	11. Pages
Various	148

12. Distribution Statement This document is distributed in accordance with AGARD policies and regulations, which are outlined on the Outside Back Covers of all AGARD publications.

13. Keywords/Descriptors

Artificial intelligence
Avionics
Control equipment
Maintenance
Guidance

14. Abstract

The use of various forms of Artificial Intelligence to help solve problems related to guidance and control implementations is a subject of current interest in the field. First applications have been of Knowledge Based methods to improved G & C equipment maintenance based on "expert knowledge" from avionics hardware experts. New programmes are in process to bring knowledge based techniques to bear in supporting the pilot in performing the tactical mission. The intent of this lecture series is to describe clearly what Artificial Intelligence techniques mean with respect to Guidance and Control applications, to offer some concrete examples of applications to the maintenance area, some more speculative examples of applications to actual G & C tasks, and a projection of possible directions for the future.

Substantial research has been performed on application of knowledge based concepts to various aspects of guidance and control systems. It is the purpose of this Lecture Series to bring some of the results of this research and accomplishment to the wider guidance and control community by providing in-depth description on principles of AI methods and results of application to real problems.

This Lecture Series, sponsored by the Guidance and Control Panel of AGARD, has been implemented by the Consultant and Exchange Programme.

AGARD Lecture Series No.155
Advisory Group for Aerospace Research and Development, NATO
KNOWLEDGE BASED CONCEPTS AND ARTIFICIAL INTELLIGENCE APPLICATIONS TO GUIDANCE AND CONTROL
Published August 1987
148 pages

The use of various forms of Artificial Intelligence to help solve problems related to guidance and control implementations is a subject of current interest in the field. First applications have been of Knowledge Based methods to improved G & C equipment maintenance based on "expert knowledge" from avionics hardware experts. New programmes are in process to bring knowledge based

P.T.O.

AGARD-LS-155

Artificial intelligence
Avionics
Control equipment
Maintenance
Guidance

AGARD Lecture Series No.155
Advisory Group for Aerospace Research and Development, NATO
KNOWLEDGE BASED CONCEPTS AND ARTIFICIAL INTELLIGENCE APPLICATIONS TO GUIDANCE AND CONTROL
Published August 1987
148 pages

The use of various forms of Artificial Intelligence to help solve problems related to guidance and control implementations is a subject of current interest in the field. First applications have been of Knowledge Based methods to improved G & C equipment maintenance based on "expert knowledge" from avionics hardware experts. New programmes are in process to bring knowledge based

P.T.O.

AGARD-LS-155

Artificial intelligence
Avionics
Control equipment
Maintenance
Guidance

AGARD Lecture Series No.155
Advisory Group for Aerospace Research and Development, NATO
KNOWLEDGE BASED CONCEPTS AND ARTIFICIAL INTELLIGENCE APPLICATIONS TO GUIDANCE AND CONTROL
Published August 1987
148 pages

The use of various forms of Artificial Intelligence to help solve problems related to guidance and control implementations is a subject of current interest in the field. First applications have been of Knowledge Based methods to improved G & C equipment maintenance based on "expert knowledge" from avionics hardware experts. New programmes are in process to bring knowledge based

P.T.O.

AGARD-LS-155

Artificial intelligence
Avionics
Control equipment
Maintenance
Guidance

AGARD Lecture Series No.155
Advisory Group for Aerospace Research and Development, NATO
KNOWLEDGE BASED CONCEPTS AND ARTIFICIAL INTELLIGENCE APPLICATIONS TO GUIDANCE AND CONTROL
Published August 1987
148 pages

The use of various forms of Artificial Intelligence to help solve problems related to guidance and control implementations is a subject of current interest in the field. First applications have been of Knowledge Based methods to improved G & C equipment maintenance based on "expert knowledge" from avionics hardware experts. New programmes are in process to bring knowledge based

P.T.O.

AGARD-LS-155

Artificial intelligence
Avionics
Control equipment
Maintenance
Guidance

techniques to bear in supporting the pilot in performing the tactical mission. The intent of this lecture series is to describe clearly what Artificial Intelligence techniques mean with respect to Guidance and Control applications, to offer some concrete examples of applications to the maintenance area, some more speculative examples of applications to actual G & C tasks, and a projection of possible directions for the future.

Substantial research has been performed on application of knowledge based concepts to various aspects of guidance and control systems. It is the purpose of this Lecture Series to bring some of the results of this research and accomplishment to the wider guidance and control community by providing in-depth description on principles of AI methods and results of application to real problems.

This Lecture Series, sponsored by the Guidance and Control Panel of AGARD, has been implemented by the Consultant and Exchange Programme. Presented on 10—11 September 1987 in Ottawa, Canada, 14—15 September 1987 in Monterey, USA, 12—13 October 1987 in Delft, The Netherlands and 15—16 October 1987 in Lisbon, Portugal.

ISBN 92-835-1588-7

techniques to bear in supporting the pilot in performing the tactical mission. The intent of this lecture series is to describe clearly what Artificial Intelligence techniques mean with respect to Guidance and Control applications, to offer some concrete examples of applications to the maintenance area, some more speculative examples of applications to actual G & C tasks, and a projection of possible directions for the future.

Substantial research has been performed on application of knowledge based concepts to various aspects of guidance and control systems. It is the purpose of this Lecture Series to bring some of the results of this research and accomplishment to the wider guidance and control community by providing in-depth description on principles of AI methods and results of application to real problems.

This Lecture Series, sponsored by the Guidance and Control Panel of AGARD, has been implemented by the Consultant and Exchange Programme. Presented on 10—11 September 1987 in Ottawa, Canada, 14—15 September 1987 in Monterey, USA, 12—13 October 1987 in Delft, The Netherlands and 15—16 October 1987 in Lisbon, Portugal.

ISBN 92-835-1588-7

techniques to bear in supporting the pilot in performing the tactical mission. The intent of this lecture series is to describe clearly what Artificial Intelligence techniques mean with respect to Guidance and Control applications, to offer some concrete examples of applications to the maintenance area, some more speculative examples of applications to actual G & C tasks, and a projection of possible directions for the future.

Substantial research has been performed on application of knowledge based concepts to various aspects of guidance and control systems. It is the purpose of this Lecture Series to bring some of the results of this research and accomplishment to the wider guidance and control community by providing in-depth description on principles of AI methods and results of application to real problems.

This Lecture Series, sponsored by the Guidance and Control Panel of AGARD, has been implemented by the Consultant and Exchange Programme. Presented on 10—11 September 1987 in Ottawa, Canada, 14—15 September 1987 in Monterey, USA, 12—13 October 1987 in Delft, The Netherlands and 15—16 October 1987 in Lisbon, Portugal.

ISBN 92-835-1588-7

techniques to bear in supporting the pilot in performing the tactical mission. The intent of this lecture series is to describe clearly what Artificial Intelligence techniques mean with respect to Guidance and Control applications, to offer some concrete examples of applications to the maintenance area, some more speculative examples of applications to actual G & C tasks, and a projection of possible directions for the future.

Substantial research has been performed on application of knowledge based concepts to various aspects of guidance and control systems. It is the purpose of this Lecture Series to bring some of the results of this research and accomplishment to the wider guidance and control community by providing in-depth description on principles of AI methods and results of application to real problems.

This Lecture Series, sponsored by the Guidance and Control Panel of AGARD, has been implemented by the Consultant and Exchange Programme. Presented on 10—11 September 1987 in Ottawa, Canada, 14—15 September 1987 in Monterey, USA, 12—13 October 1987 in Delft, The Netherlands and 15—16 October 1987 in Lisbon, Portugal.

ISBN 92-835-1588-7

Printed by Specialised Printing Services Limited
40 Chigwell Lane, Loughton, Essex IG10 3TZ

ISBN 92-835-1588-7